

Jan Ramon
ECML 2013

LEARNING AND MINING WITH NETWORK-STRUCTURED DATA



Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
- Learning

Introduction

Network = Objects + Relations

| Application | Objects | Relations |
|---------------------------------|--------------------------|---------------------------------|
| Social network | Person | Friendship, colleague |
| Traffic network | Crossroad | Road |
| Chemical interaction net | Chemicals | Interaction |
| Telecommunication net | Person | Phone call |
| Citation network | Paper, researcher | Citation, authorship |
| Shareholder network | Company, Person | Shareholdership |
| Computer network | Computer, router | Cable, wifi registration |

Introduction

- In this tutorial:
 - Who studies networks?
 - Network patterns & mining them
 - Learning in networks
- Focus on
 - Local patterns
 - Not so much on large scale properties

Related ECML/PKDD 2013 tutorials

- [Fri-PM] Algorithmic techniques for modeling and mining large graphs (Alan Frieze et al.)
 - Focus is more on global properties
- [Mon-PM] Discovering Roles and Anomalies in Graphs: Theory & Applications (T. Eliassi-Rad et al.)
 - Anomaly detection is not covered here
- [Fri-AM] Statistically sound pattern discovery (G. Webb & W. Hamalainen)
 - Different statistical aspects

Introduction

Prerequisites

Supervised learning: Given i.i.d. training examples, learn a function from example to target value.

- ▣ You could use SVM, DT, NB, IBL, GP, ... or any of your favorite supervised techniques

Contents

- Introduction
- **Networks from different points of view**
- Patterns & pattern mining
- Learning

Contents

- Introduction
- Networks from different points of view
 - Basic concepts
 - Data mining tasks
 - Relevant fields of research
- Patterns & pattern mining
- Learning

Basic concepts - Graphs

- An **undirected (labeled) graph** is a tuple $G=(V,E,\lambda)$ where
 - V is a set of **vertices (nodes)** [punten (knopen)]
 - $E \subseteq \{\{v,w\} | v,w \in V\}$ is a set of **edges** [takken]
 - $\lambda: V \cup E \rightarrow \Sigma$ is a **labeling function**
- **Unlabeled graph:**
 - If λ is constant (all vertices/edges have the same label), λ may be omitted

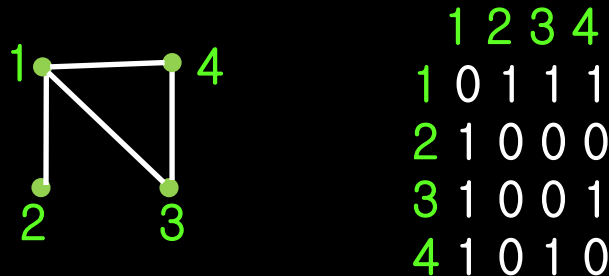
Basic concepts - Graphs

- A **directed graph** is a tuple $G=(V,E,\lambda)$ where
 - V is a set of vertices (nodes) [punten (knopen)]
 - $E \subseteq \{(v,w) | v,w \in V\}$ is a set of **arcs** [bogen]
 - $\lambda: V \cup E \rightarrow \Sigma$ is a labeling function
- Further notations:
 - $V(G)$ is the set of vertices of the graph G
 - $E(G)$ is the set of edges / arcs of the graph G
 - λ_G is the labeling function of the graph G
 - $N_H(v) = \{w \in V(H) | \{v,w\} \in E(H)\}$ is the neighborhood of v
 - $\Delta v = N_H(v)$ is the degree of v

Basic concepts

adjacency matrix

- **Adjacency matrix** of graph G is a square matrix A of dimension $V(G) \times V(G)$ such that
 - $A_{u,v} = 0$ if u and v are not connected
 - $A_{u,v} = 1$ if there is an edge between u and v



Basic concepts – Walk/Path

- A **walk** P between vertices v and w in a graph G is a sequence of vertices $u_1, u_2, \dots, u_n \in V(G)$ such that
 - $u_1 = v$,
 - $u_n = w$ and
 - $(u_i, u_{i+1}) \in E(G)$ for all $1 \leq i \leq n - 1$.
- The **length** of such walk P is $n - 1$.
- A **path** is a walk where all vertices are distinct
- Slightly abusing terminology, a path P can also be seen as a subgraph of G

Basic concepts – Shortest path

- A **shortest path** is a path of minimal length.
- **Distance** $d(u, v)$ between u and v is length of shortest path between u and v
- The **diameter** of G is

$$\text{diam}(G) = \max_{u, v \in V(G)} d(u, v)$$

Basic concepts – Diameter

The **diameter** of G is

$$\text{diam}(G) = \max_{u,v \in V(G)} d(u, v)$$

Many real-world graphs have small diameter.

- V : all persons
- E : an edge connects persons who have ever met each other
- Many people have met a local politician who met the national prime minister

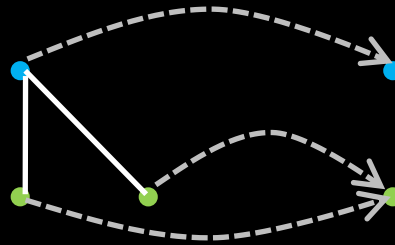
Basic concepts – connected, tree

- A graph G is **connected** iff there is a path between every pair of vertices $v, w \in V(G)$
- A **connected component** of a graph G is a maximal connected subgraph of G .
- A graph G is a **tree** iff there is a unique path between every pair of vertices $v, w \in V(G)$
 - ▣ Intuition: if the path between two vertices is not unique, then there is a cycle.

Basic concepts – morphisms

- A **homomorphism** from a graph H into a graph G is a mapping $\varphi: V(H) \rightarrow V(G)$ such that
 - $\forall v, w \in V(H): (v, w) \in E(H) \Rightarrow (\varphi(v), \varphi(w)) \in E(G)$
 - $\forall v \in V(H): \lambda(v) = \lambda(\varphi(v))$
 - $\forall v, w \in V(H): \lambda(v, w) = \lambda(\varphi(v), \varphi(w))$
- An injective homomorphism is a **subgraph isomorphism**.

Basic concepts – subgraph isomorphism vs homomorphism



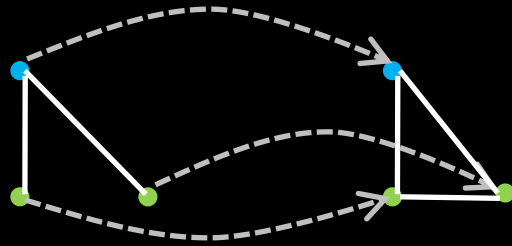
A homomorphism, not an isomorphism

- If there is a homomorphism from H to G , then we denote this $H \leq_h G$
- If there is a subgraph isomorphism from H to G , then we denote this $H \leq_i G$
- $H \equiv G$ iff $H \leq G$ and $G \leq H$

Basic concepts – morphisms

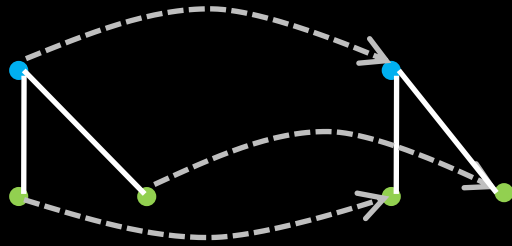
- An **induced homomorphism** from a graph H into a graph G is a mapping $\varphi: V(H) \rightarrow V(G)$ such that
 - $\forall v, w \in V(H): (v, w) \in E(H) \Leftrightarrow (\varphi(v), \varphi(w)) \in E(G)$
 - $\forall v \in V(H): \lambda(v) = \lambda(\varphi(v))$
 - $\forall v, w \in V(H): \lambda(v, w) = \lambda(\varphi(v), \varphi(w))$
- An injective induced homomorphism is an **induced subgraph isomorphism**.

Basic concepts – induced vs normal subgraph isomorphism



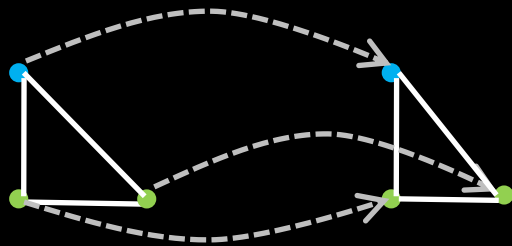
Subgraph isomorphism

NOT induced subgraph isomorphism



Subgraph isomorphism

Induced subgraph isomorphism



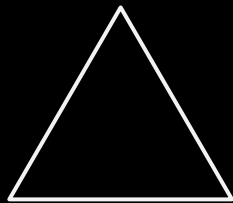
Subgraph isomorphism

Induced subgraph isomorphism

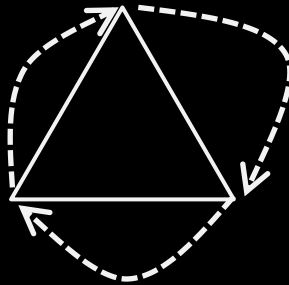
Basic concepts

Automorphisms

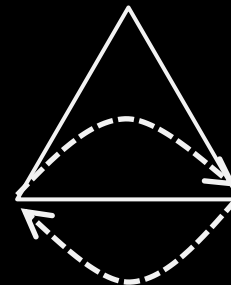
- **Automorphism** = isomorphism of a graph on itself.
- $|aut(H)|$ is the size of the **automorphism group** $aut(H)$.
- For bounded degree H , one can compute $|aut(H)|$ in polynomial time.



Triangle
“Mirror”

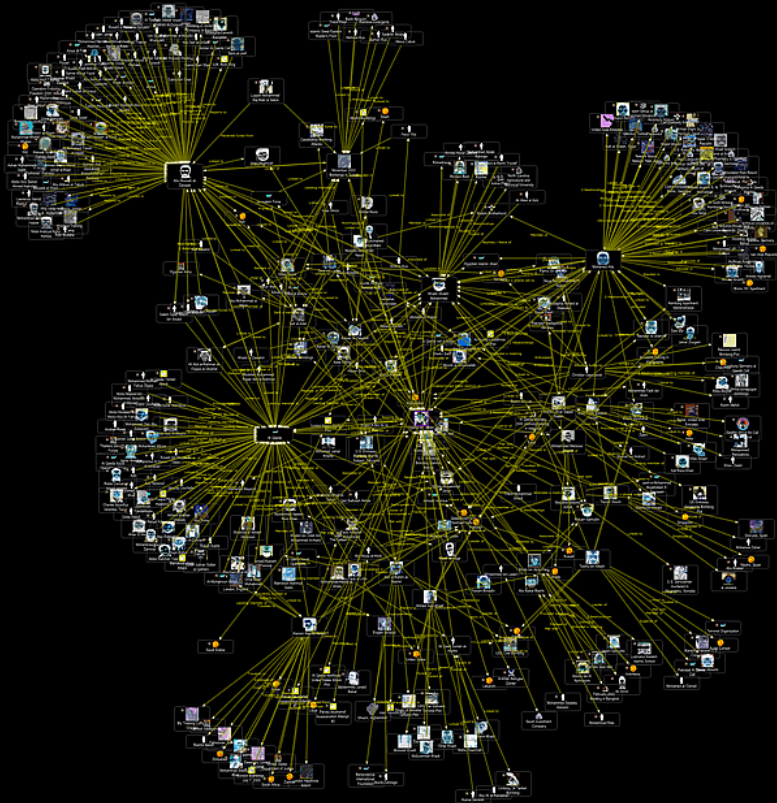


“Rotation”

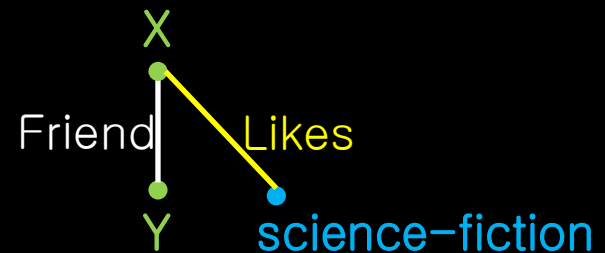


$Aut(\text{triangle})$ has size $2 \cdot 3 = 6$

Basic concepts - Networks



Network =
big database graph



Pattern =
small graph

Contents

- Introduction
- Networks from different points of view
 - Basic concepts
 - **Data mining tasks**
 - Relevant fields of research
- Patterns & pattern mining
- Learning

Network data mining tasks

| | Global / asymptotic | Local |
|-----------|---|--|
| Static | 1. Clustering & community detection | 2. Pattern mining 3. Edge/vertex structure/feature learning |
| Evolution | 5. Generative models | 4. Temporal learning |

Clustering & community detection

- Given:

- network D

- Find:

- Set of subsets (clusters, communities) V_1, V_2, \dots, V_n of $V(G)$
 - Covering $V(G)$ or not
 - Disjoint or overlapping
 - such that vertices in same cluster are close
 - similar or connected
 - and vertices from different clusters are distant
 - dissimilar / not connected

| | Global / asymptotic | Local |
|-----------|---------------------|-------|
| Static | | |
| Evolution | | |

Clusters & communities examples

- Find groups of people who are densely connected
- Find groups of people who have a similar opinion or behavior (and are connected)
- People in the same country, company, school, domain, ... will often cluster together.

Pattern mining

- Given:

- network D
- pattern language L
- interestingness criterion $I: L \times D \rightarrow \{true, false\}$

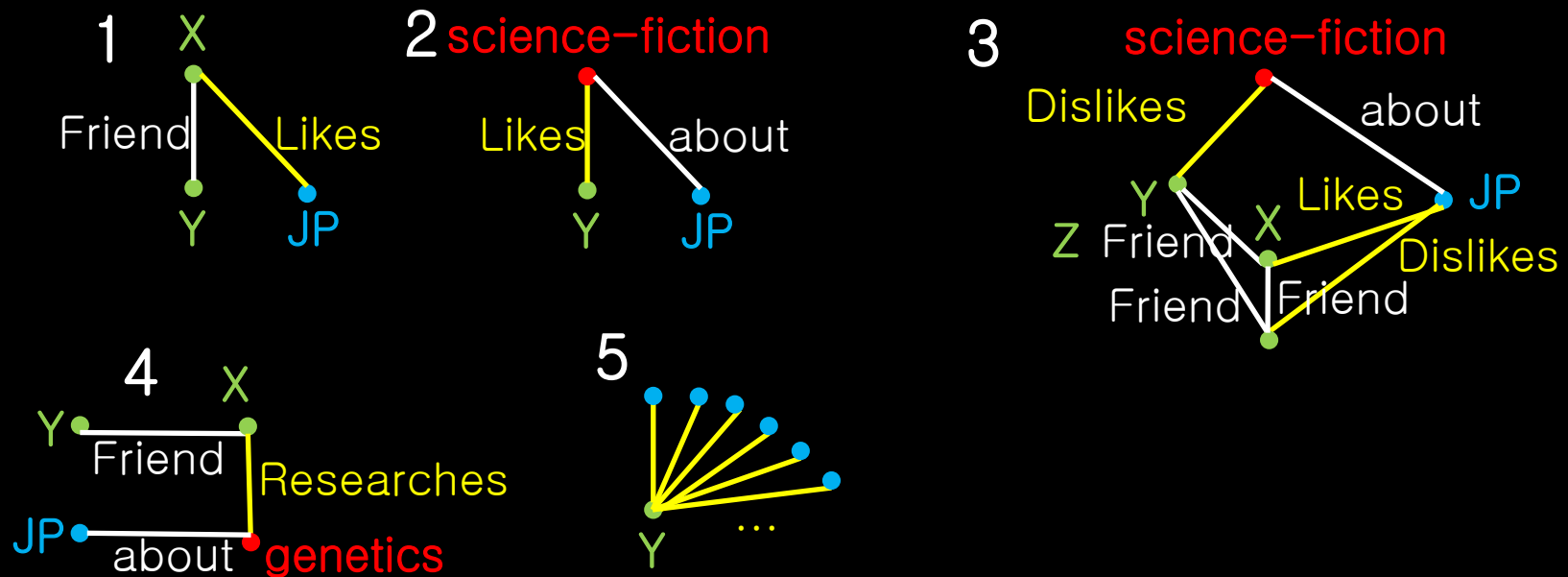
- Find

- all patterns $P \in L$ for which $I(P, D)$

| | Global / asymptotic | Local |
|-----------|------------------------|-------|
| Static | | |
| Evolution | | |

Pattern mining example

Why do you (Y) watch “Jurassic park” (JP)?



Vertex/edge structure/ feature prediction

- Given:

- network D ,
- example set $X = V(D) \cup E(D)$, target space Y
- Unknown distribution P on $X \times Y$
- Training set $Z_{train} \subseteq X \times Y$
- Test set $X_{test} \subseteq X$
- Loss function $L: Y \rightarrow \mathbb{R}^+$

- Find

- \hat{f} minimizing $E[\sum_i L(\hat{f}(x_i^{test}), y_i^{test})]$

| | Global | Local |
|-----------|--------|-------|
| Static | | |
| Evolution | | |

vertex/edge structure/ feature prediction example

- Predicting on existing objects:
 - Social network: Given a user, his profile, his friendship relations, is this user interested in chess?
 - Given a pair of friends (X,Y). Is X planning to send a message to Y today?
- Predicting on hypothetical objects:
 - Given a group of people, do they have a common friend (not yet in the network) ?
 - Given two people. do they know each other (even though not yet represented in the network)?

Learning from temporal data

| | Global / asymptotic | Local |
|-----------|---------------------|-------|
| Static | | |
| Evolution | | |

- Given:

- time-dependent network $\{D_t\}_{t=1}^T$, (possibly represented by vertices and edges with time stamps etc.)
- a loss function $L: \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}^+$

- Find:

- Prediction $\hat{D}_{T+\Delta t}$ of the network (or parts thereof) to minimize $\mathbb{E}[L(D_{T+\Delta t}, \hat{D}_{T+\Delta t})]$

Learning from temporal data example

- Social network:
 - When will X update his profile?
 - Will X and Y become friends?
 - Will a common friend of X and Y join the network?

Learning generative models

| | Global / asymptotic | Local |
|-----------|---------------------|-------|
| Static | | |
| Evolution | | |

- Generative model = probability distribution mapping a network on a new network, i.e.
- $h: \mathcal{G} \times \mathcal{G} \rightarrow [0,1]$ s.t.
 $\forall D \in \mathcal{G}, \sum_{D' \in \mathcal{G}} h(D, D') = 1$

Learning generative models

- $h: \mathcal{G} \times \mathcal{G} \rightarrow [0,1]$
- Given:
 - A hypothesis space \mathcal{H} of generative models
 - An unknown $h \in \mathcal{H}$, we assume there was some D_0 and for $t = 1..T$, D_t was drawn from $h(D_{t-1}, \cdot)$
 - time-dependent network $\{D_t\}_{t=1}^T$
- Find:
 - Given loss function $L: \mathcal{H} \times \mathcal{H} \rightarrow \mathbb{R}^+$, a model $\hat{h} \in \mathcal{H}$ such that $E[L(h, \hat{h})]$ is minimal
 - Model $\hat{h} \in \mathcal{H}$ such that $h^n(D_0)$ has the same asymptotic properties as D_T .

Learning generative models

- Difference with 'learning from temporal data':
 - Not just predicting a future event, but also global properties should be right
 - E.g. errors which propagate quickly should be avoided. (e.g. positive feedback loops)

What data do I need?

- Many datasets are undirected unlabeled graphs (interaction = yes/no)
 - Ok for many models focussing on global & asymptotic aspects
- How about correlations between interests of friends?
 - Then you need to records people's interests
 - In general, "local" models will need richer data

Contents

- Introduction
- Networks from different points of view
 - Basic concepts
 - Data mining tasks
 - Relevant fields of research
- Patterns & pattern mining
- Learning

Relevant fields of research

- Statistical physics
- Complex systems
- Multi agent systems, ants, other simulation
- Grammar induction
- (Algorithmic) graph theory
- Spectral graph theory
- pattern mining, data mining, machine learning

Statistical physics

- If we assume a given set of laws, what will happen?
- Law = graph generation model
 - Erdos-Reny model:
 - Barabasi-Albert model
- Result =
 - Asymptotic behavior, what happens if there are *many* particles?

Statistical physics

The Erdos-Reny model

- A **random graph** from the **Erdos-Reny** distribution $G_p(n,p)$ is constructed as follows:
 - Let G be a graph on n vertices.
 - For every pair of vertices $\{v,w\}$, connect v and w with an edge with probability p .
- A random graph from the Erdos-Reny distribution $G_M(n,M)$ is constructed as follows:
 - Let G be a graph on n vertices.
 - Choose randomly M elements from $\{\{v,w\} \mid v,w \in V(G)\}$ and draw an edge between the two elements of these pairs.

Statistical physics

Allmost all graphs

- Let G_n be a random graph drawn from $G(n, p(n))$, i.e. p is a function of n . A predicate q (i.e. a boolean function) holds for **(asymptotically) allmost surely (a.a.s.)** if

$$\lim_{n \rightarrow \infty} P(q(G_n) = \text{true}) = 1$$

- Similar for $G(n, M)$
- If no $G(n, p)$ or $G(n, M)$ specified: $G(n, 1/2)$ by default (“allmost every graph”).

Statistical physics

Asymptotic properties

- E.g. the “giant component”
 - If $\lim_{n \rightarrow \infty} np < 1$, then the largest component of a $G(n, p)$ graph is a.a.s. not larger than $3 \cdot \log(n) / (1 - np)^2$
 - If $\lim_{n \rightarrow \infty} np = 1$, then the largest component of a $G(n, p)$ graph is a.a.s. $n^{2/3}$
 - If $\lim_{n \rightarrow \infty} np > 1$, then the largest component of a $G(n, p)$ graph is a.a.s. close to βn with $\beta + e^{-\beta p n} = 1$

Statistical physics

Asymptotic properties

- E.g. connectedness:
 - If $\lim_{n \rightarrow \infty} np/\ln(n) < 1$ then, $G(n,p)$ is a.a.s. disconnected
 - If $\lim_{n \rightarrow \infty} np/\ln(n) > 1$ then, $G(n,p)$ is a.a.s. connected

Statistical physics

0-1 law

- Given a first order logic formula F over graphs, $\lim_{n \rightarrow \infty} P(F(G(n, 1/2)) = \text{true})$ is either 1 or 0.
- E.g. “contains a triangle”: adding vertices (and hence edges) only increases the probability of a triangle; if many vertices, the probability gets close to 1.

Statistical physics

What can ML and DM learn?

- Compute from the model the “expected value” of a pattern.
- An “interesting pattern” is one which deviates from the expected value according to the model.
- E.g. assume G is the union of a random graph and a clique. Under what conditions can we detect the clique as an abnormally dense spot?¹

¹ E.g. “Detecting bicliques in $GF[q]$. ECML2013”

Complex systems

- Model processes in society
 - systems of interacting individuals
 - what (often large-scale) properties do we observe?
- Study behavior of systems with such properties
 - asymptotics
 - simulation of systems
 - emerging patterns

Complex systems

What can ML/DM learn?

- Techniques to model application domains
 - Social behavior
 - Economics
 - ...

Multi-agent systems, ants, simulation

- How to simulate?
- Do artificial populations offer more value than artificial individuals?

Multi-agent systems

What can ML/DM learn?

- Simulation (sampling a temporal model forward in time)
- Multi-agent learning (and the effects of changing behavior due to learning)
- Game theory (~statistical physics: Nash equilibria)

Grammar induction

- Generative model ~ probabilistic grammar
 - Initial state
 - (probabilistic) production rules
- Graph grammars
 - Hyperedge replacement grammars
 - Vertex replace grammars

Grammar induction

What can DM/ML learn?

- Graph grammars: learning generative models producing certain probability distributions over graphs

(Algorithmic) graph theory

- Algorithms on graphs and their complexity are well-studied.
 - **How** to solve graph problems
 - **Complexity** of solving problems

(Algorithmic) graph theory

What can ML/DM learn?

- Pattern matching
 - ▣ E.g. see part 2
- Support measures
 - ▣ e.g. Maximum independent set, Lovasz theta function, ... (part 3)
- Shortest path algorithms
- Similarity, maximum common subgraph, ...
- ...

Relational databases

- A network is a relational database where the binary “edge” relation has foreign keys to itself.
- Matching patterns = evaluating queries

Relational databases

What can ML/DM learn?

- Database theory sometimes shows how to match patterns (=evaluate queries), but is usually not optimized for “recursive” foreign keys.
- Many ideas for data structures (e.g. recently growing interest in graph database indexing)

Spectral graph theory¹

- Study of
 - the adjacency matrix of a graph
 - its eigenvalues
 - the Laplacian matrix: $L = D - A$ with D degree matrix (D_{vv} is degree of v) and A adjacency matrix.
 - ...

¹ Fan Chung, “Spectral graph theory”

Spectral graph theory

What can ML/DM learn?

- Laplacian describes “influence flow”, used in
 - semi-supervised learning
 - manifold embedding
 - ...
- Clustering:
 - # of zero eigenvalues = # of connected components

Mining, learning

- Relational learning (e.g. SRL)
- Learning in graphs (e.g. MLG)
- Using logical representations (e.g. ILP)
 - (but adding logic makes problems often undecidable)

Introduction summary (1/3)

Basic concepts

- Graphs
 - labels
 - adjacency matrix
- Paths
 - distance
- Morphisms (matching operators)

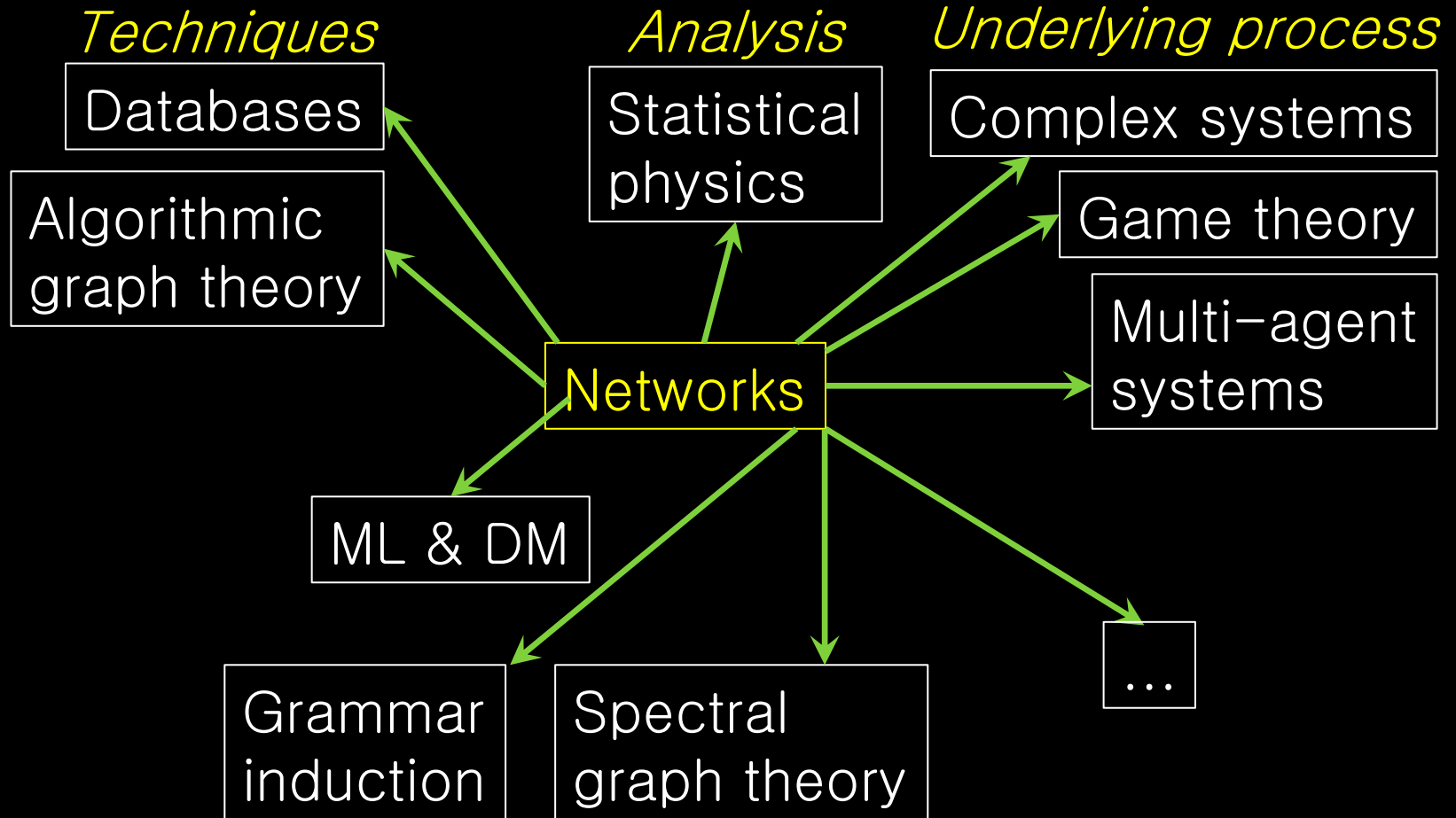
Introduction summary (2/3)

Network data mining tasks

| | Global / asymptotic | Local |
|-----------|---|--|
| Static | 1. Clustering & community detection | 2. Pattern mining 3. Edge/vertex structure/feature learning |
| Evolution | 5. Generative models | 4. Temporal learning |

Introduction summary (3/3)

Domains researching networks



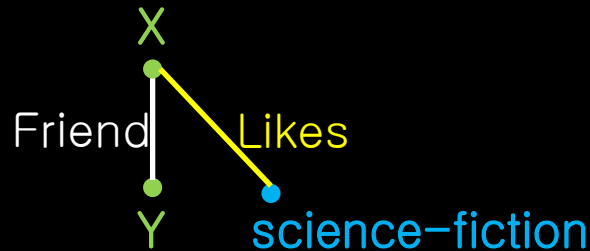
Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
 - Introduction
 - Pattern matching
 - Frequency
 - Additional remarks
- Learning

Introduction

What is a pattern?

- Pattern = collection of vertices that should satisfy some constraints (connections, labels, ...)



Introduction

A generic pattern miner

Assume *interesting* is anti-monotonic

$k \leftarrow 0$; $C_0 \leftarrow \text{MinimalPatterns}$

While $C_k \neq \{\}$

$S_k \leftarrow \{P \in C_k \mid \text{interesting}(P)\}$

$C_{k+1} \leftarrow \bigcup_{P \in S_k} \text{extensions}(P)$

$k \leftarrow k + 1$

EndWhile

$Solutions \leftarrow \bigcup_k S_k$

Introduction

A generic pattern miner

- “assume *interesting* is anti-monotonic” allows for pruning.

$$\begin{aligned} & \textit{interesting}(G) \wedge G \leq H \\ & \implies \textit{interesting}(H) \end{aligned}$$

- Can also mine using non-anti-monotonic criterion (e.g. correlated patterns¹)
- Breadth-first “Apriori-style” ²
 - Also depth-first possible ³
- To be instantiated:
 - *MinimalPatterns & Extensions*
 - *Interesting*

¹ Zimmermann & DeRaedt, DS2004

² Agrawal & Srikant, VLDB1994

³ Han, Pei, Yin SIGMOD2000

Introduction

Enumeration of patterns

To be instantiated:

- *MinimalPatterns* & *Extensions*
- *Interesting*

- Many approaches are generate-and-test
- How to generate all graphs subject to anti-monotonic constraint?
 - Practice: mining graph patterns in databases of transactions represented with graphs: AGM, gSpan, FSG, Gaston² → use canonical form
 - Theory: polynomial-delay (+evaluation anti-monotone criterion)¹

¹ Ramon & Nijssen, JMLR2008

² Nijssen & Kok 2004

Introduction

Complexity notions

- Enumeration/**listing problems** (“find all ...”) may have output O exponential in input size I .
- **Polynomial delay**: between solution $j - 1$ and j at most $\text{poly}(I)$ time.
- **Incremental polynomial time**: between solution $j - 1$ and j at most $\text{poly}(I, j)$ time.
- **Output-polynomial time**: total running time at most $\text{poly}(I, O)$

Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
 - Introduction
 - **Pattern matching**
 - Frequency
 - Additional remarks
- Learning

To be instantiated:

- *MinimalPatterns & Extensions*
- *Interesting*

Pattern matching

Overview

- Problem statement
- Hardness results
- Triangle counting
- Small patterns
- Larger patterns
 - Cliques
 - Sampling
 - Fixed parameter tractability

Pattern matching

The problem

- Given:

- A network D
- A pattern P

- Find

- (all/some/one/...) embeddings of P in D OR
 - an aggregate (count, average, ...) computed over these embeddings
-
- The diagram shows two labels, 'listing problem' and 'decision problem', in italics. Two arrows originate from these labels. The arrow from 'listing problem' points to the first option of the 'Find' section: '(all/some/one/...) embeddings of P in D OR'. The arrow from 'decision problem' points to the same option.

Pattern matching

Why do we care?

- Basic operation for both learning and mining
- There is a literature on basic pattern matching, but learning & mining queries have specific characteristics
 - Data is rich, satisfies integrity constraints, ...
 - Patterns may have wildcards
 - DM/ML is aimed at collecting statistics

Pattern matching

Subgraph isomorphism complexity

- Pattern P , network D
- List embeddings $\pi: P \rightarrow D$
- $\#P$ -complete
- Classic algorithms (backtracking search):

$$O(|V(D)|^{|V(P)|})$$

$\#P$: counting problems f such that there is a polynomial-time non-deterministic Turing machine for which upon input x the number of accepting states equals $f(x)$. $\#P \supseteq NP$

Pattern matching

Heuristic search

Ullmann's algorithm¹ (pattern P , network D)
 $\text{Match}(P, D, \{\})$

Procedure $\text{Match}(P, D, \text{partial embedding } \pi)$
 If $V(P) = \text{dom}(\pi)$
 then $\text{ListSolution}(\pi)$
 else
 Select $v \in V(P) \setminus \text{dom}(\pi)$
 Let $C = \{w \in V(D) \mid w \text{ maybe image of } v\}$
 For all $w \in C$ do
 $\text{Match}(P, D, \pi \cup \{(v, w)\})$

¹ Ullmann, JACM 23(1), 1976

Pattern matching

Avoiding worst-case complexity

- Database of transactions of graphs
 - E.g. many small molecule graphs, RNA
 - exploit structure of database graphs, e.g.
 - atoms have max degree = 6 ¹
 - molecules are often planar (or even outerplanar ²)
 - bounded treewidth ³

¹ E.M.Luks, J Computer & System Sciences 25(1), 1982

² Horvath & Ramon, DMKD 21(3), 2010

³ Horvath & Ramon, TCS 411, 2010

Pattern matching

Avoiding worst-case complexity

- Network
 - ✗ No definite structure we can rely on
 - ✗ Approximate matching may help but
 - Subgraph isomorphism is hard to approximate
 - ✓ Structural properties of patterns
 - Triangles and other small patterns
 - Trees
 - ✓ Statistical properties of network
 - Random graphs

Pattern matching

Triangle counting

- Simple problem:
 - Given: a triangle P and a network D
 - List or count: all embeddings of P in D
- but a lot of literature on solving it

Pattern matching

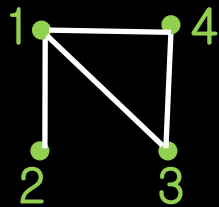
Triangle counting

- Idea 1: Brute force
 - Check every triple of vertices of G
 - Runtime $O(|V(G)|^3)$

Pattern matching

Triangle counting

- Idea 2: matrix multiplication ¹
 - Let A be the adjacency matrix of G
 - $(A^n)_{u,v} = \#$ of walks of length n between u and v
 - Matrix multiplication is $O(|V(G)|^{2.37})$ ²
 - Hence, compute $\text{tr}(A^3)/3!$



$$A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$A^2 = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}$$

$$A^3 = \begin{pmatrix} 2 & 3 & 4 & 4 \\ 3 & 0 & 1 & 1 \\ 4 & 1 & 2 & 3 \\ 4 & 1 & 3 & 2 \end{pmatrix}$$

$$\text{tr}(A^3)/3! = (2+2+2)/6 = 1$$

¹ Alon et al. 1997

² For practical problems, the exponent will be higher

Pattern matching

Triangle counting

- Idea 3: Sparse graphs
 - Iterate over edges¹: $O\left(|E(G)|^{\frac{3}{2}}\right)$
 - Nodelerator: Iterate over pairs of neighbors of vertices: $O(d_{max}^2 |E(G)|)$ with d_{max} the maximal degree of G .

¹ Itai and Rodeh 1978

Pattern matching

Triangle counting

- Idea 4: Approximation
 - Sparsification: delete randomly part of the graph, count triangles, adjust for sampling
 - Sampling: perform a number of random attempts, adjust for sampling
 - Partition graph into parts, solve them, adjust for triangles spanning several partitions.

Pattern matching

Triangle counting

| | | |
|-----------------------------------|--------|--------------------------------------|
| Alon et al., 1997 | Exact | $O(V(G) ^{2.37})$ |
| Itai & Radeh, 1978 | Exact | $O\left(E(G) ^{\frac{3}{2}}\right)$ |
| Tsourakakis ICDM2008 | Exact | |
| Faloutsos KDD2009 | Approx | Sparsification |
| Avron 2010 | Approx | Sampling |
| Pagh&Tsourakakis, InfProcLet 2012 | Approx | Partitioning |
| Becchetti et al. TKDD 2010 | Approx | |

¹ Itai and Rodeh 1978

Pattern matching

Counting motifs of size 3-5

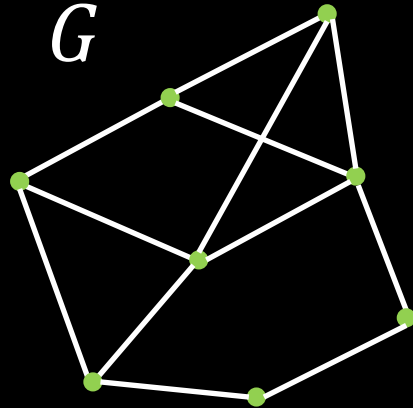
- Motifs = not necessarily connected induced subgraph
- Brute force sometimes still works¹
- Sampling strategy (e.g. GUISE²):
 - Consider an induced graph
 - Omit vertex or add neighbor
 - Metropolis hastings: adjust for fact that higher-degree vertices are reached more easily

¹ Shen–Orr et al. Nat. Genet.2002

² Bhuiyan et al. ICDM 2012

Pattern matching

Sampling motifs with MCMC

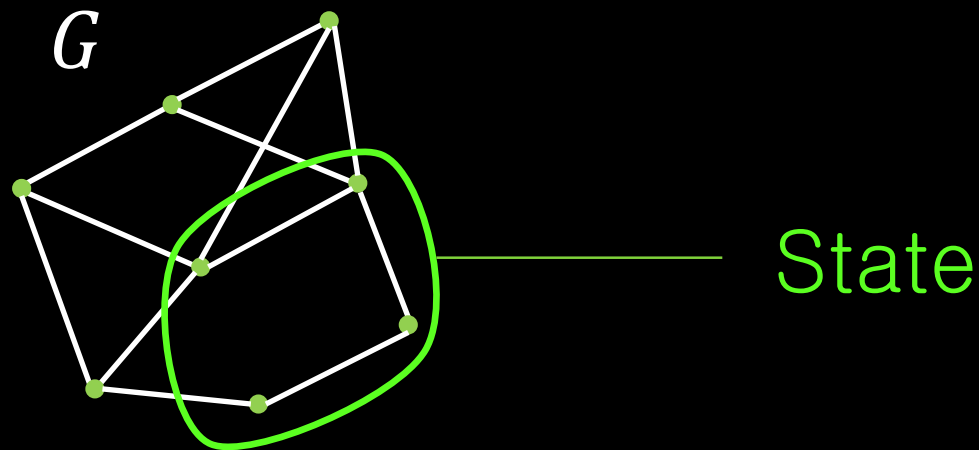


Motifs

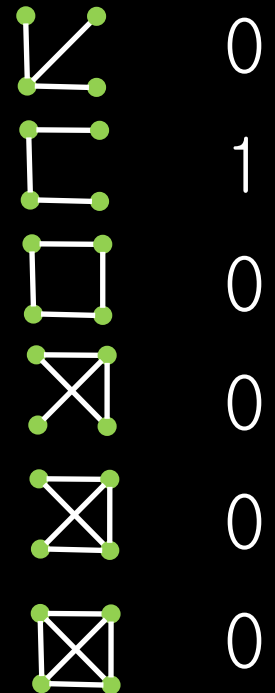


Pattern matching

Sampling motifs with MCMC

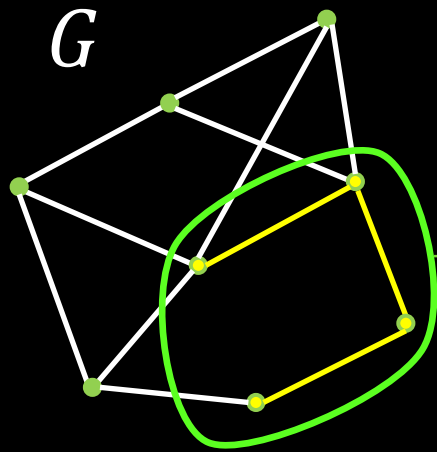


Motifs



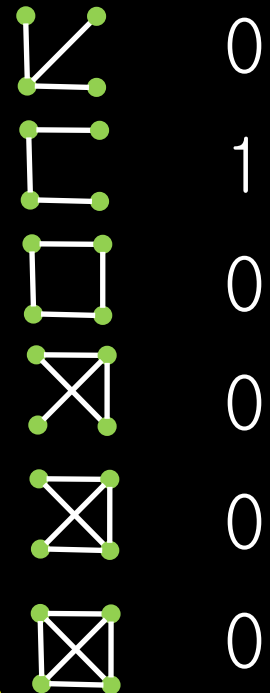
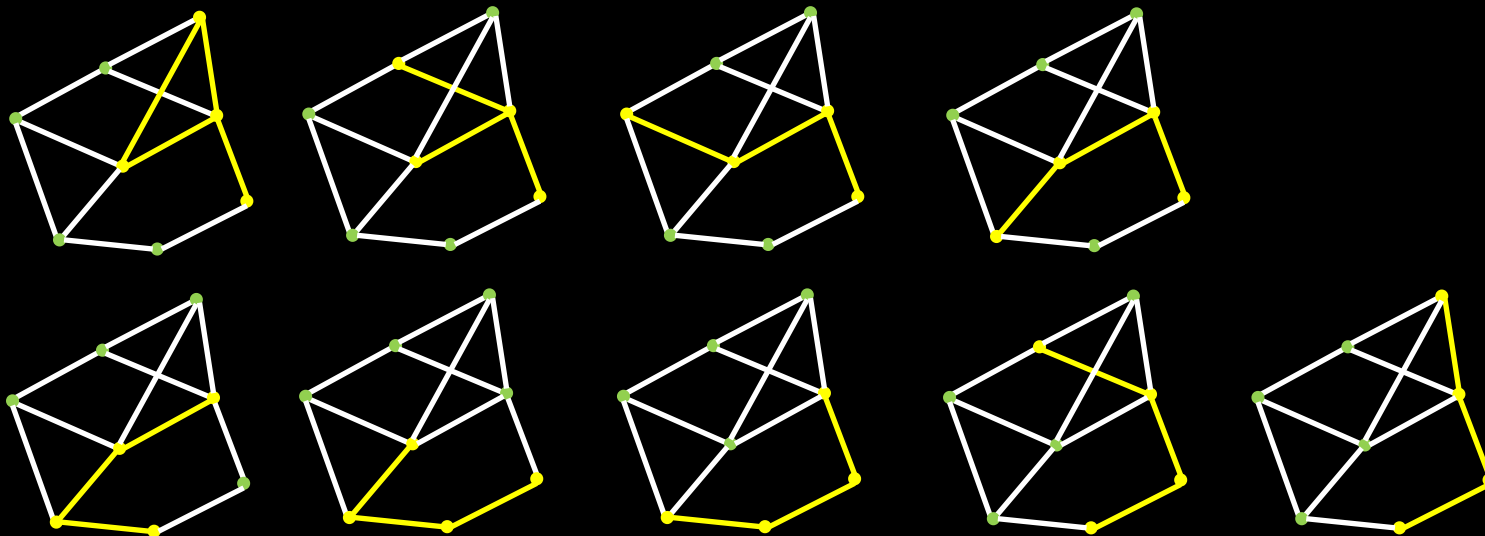
Pattern matching

Sampling motifs with MCMC



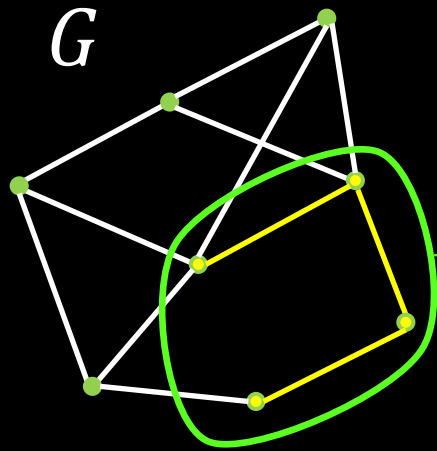
State s_1
 $d(s_1)=9$

Motifs



Pattern matching

Sampling motifs with MCMC

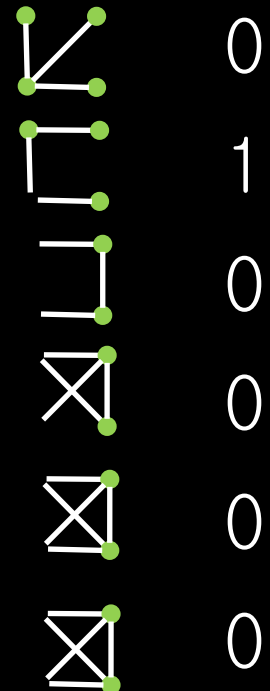


State s1
 $d(s1)=9$

$$T(u, v) = \min \left(\frac{1}{d(u)}, \frac{1}{d(v)} \right) \quad (\text{for } u \neq v)$$

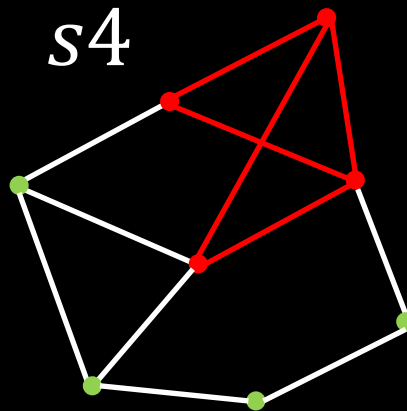
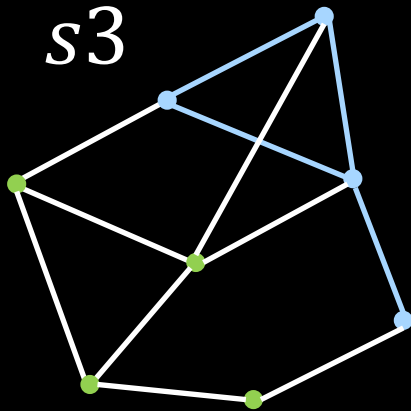
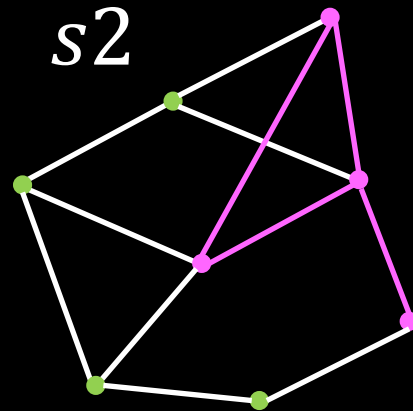
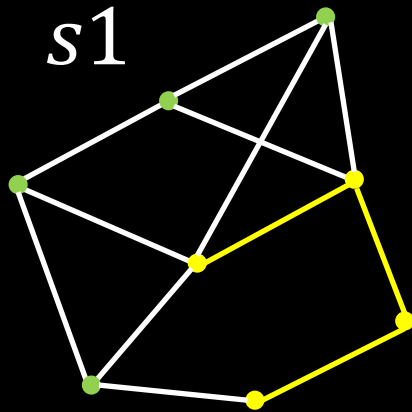
$$T(u, u) = 1 - \sum_{v \neq u} T(u, v)$$

Motifs



Pattern matching

Sampling motifs with MCMC



Motifs



1

1

0

1

1

0

Pattern matching

Overview

- Problem statement
- Hardness results
- Triangle counting
- Small patterns
- Larger patterns
 - Cliques
 - Sampling
 - Fixed parameter tractability

Pattern matching

Cliques

- Several approaches aim at finding maximal cliques
- Clique detection for non-directed graphs,

Clusters & communities

Bicliques

- Bi-partite networks $D = (V_1 \cup V_2, E)$ with $E \subseteq V_1 \times V_2$.
- Bi-clique = $C_1 \times C_2$ with $C_1 \subseteq V_1$ and $C_2 \subseteq V_2$.
- Several variants:
 - Tile mining (in boolean matrices, unweighted networks)
 - Non-negative matrix factorization (real-valued matrices, weighted networks)

Advertisement:

Join ECML/PKDD2013 Tue1B session on Networks (1)

J.Ramon, P.Miettinen, J.Vreeken, Detecting bicliques in $\text{GF}[q]$,

Pattern matching

Larger patterns: Sampling

Pattern matching

Avoiding worst-case complexity

- Network

- ✗ No definite structure we can rely on

- ✗ Approximate matching may help but

- Subgraph isomorphism is hard to approximate

- ✓ Structural properties of patterns

- Triangles and other small patterns **DONE**

- Trees

- ✓ Statistical properties of network

- Random graphs

Sampling



Pattern matching

Sampling – strategies

- Partitioning
 - Match pattern in one or all partitions
 - Assume that network has rather uniform structure
 - Scale result to full network
 - Worked for triangle counting, harder for larger patterns.
- Simulation (e.g. Fürer)
 - Attempt several times to find single match
 - assume network is sufficiently uniform
 - average over iterations

Fürer's algorithm¹ for pattern matching in random graphs

- Input:
 - Network G drawn from $G(n,p)$
 - Pattern H with a decomposition (see later)
- Output:
 - $|Emb(H, G)|$, the number of images of H in G .
- Complexity:
 - Exact & worst case: #P-complete
 - Exact for “most graphs”: still untractable
 - Approximate for most graphs : this algorithm (FPRAS)

¹ Martin Fürer & Shiva Prasad Kasiviswanathan Approx/Random 2008

Pattern matching

Fürer's algorithm - FPRAS

- Fully Polynomial Randomized Approximation Schema:
 - Randomized algorithms outputting for almost every graph in polynomial time a solution with relative error of ε .
- A little more formally:
 - An FPRAS is a randomized algorithm such that there is a polynomial $p(\cdot, \cdot)$ such that for every δ there is an n_0 such that for all $n > n_0$ and ε , and for at least a fraction $1 - \delta$ of the graphs of size n , the algorithm outputs in time at most $p(n, 1/\varepsilon)$ a solution within a factor $1 \pm \varepsilon$ of the correct solution.

Pattern matching

Fürer: Basic algorithm

```
Function unbiased_estimator(H,G)    [estimates  $|Emb(H,G)|$ ]  
    ( $\varphi, P(\varphi)$ ) = Try_to_find_embedding(H,G);  
    IF  $\varphi$  = failed  
        THEN return 0; [No embedding found]  
        ELSE return  $1/P(\varphi)$ ; [return inverse  
                                probability of  $\varphi$ ]  
EndFunction
```

```
Function count_embeddings(H,G, $\varepsilon$ )  
     $c = 0$ ;  $s = C/\varepsilon^2$ ;  
    For  $i = 1 \dots s$  do  $c = c + \text{unbiased\_estimator}(H,G)$ ; EndFor  
    Return  $c/s$ ;  
EndFunction
```

Pattern matching

Fürer: Basic algorithm

- Count_embeddings works correctly:
 - Every embedding φ is found with probability $P(\varphi)$ by Try_to_find_embeddings(H,G). With probability $1 - \sum_{\varphi} P(\varphi)$ Try_to_find_embeddings fails
 - Every embedding φ , is found once in $1/P(\varphi)$ calls, and in that case, unbiased_estimator returns $1/P(\varphi)$. Hence, φ contributes 1 to each call to unbiased_estimator, on average.
 - Hence, on average, unbiased_estimator returns the number of embeddings
- Task left: find a good Try_to_find_embedding

Pattern matching

Fürer - strategy

- Decompose vertex set of pattern
- match one partition at a time until complete
- Compute probability of finding this particular solution
- Show the overall algorithm converges sufficiently fast.
 - The sum of a (very) large number of identical distributions becomes Gaussian. Standard deviation goes down with square of sample
 - Whatever the sample size, we can always (with very small probability) have a large error

Pattern matching

Fürer: finding embeddings

INPUT: G, H , partition $\{V_1, V_2, \dots, V_l\}$ of $V(G)$

$X \leftarrow 1; \varphi_0 \leftarrow \{\}$

For $i = 1..l$

$E \leftarrow \{\varphi \in Emb(\cup_{j=1}^i V_j, G) \mid \varphi \supset \varphi_{i-1}\}$

$X_i \leftarrow |E|$

if $X_i = 0$ then terminate and return (failed, 0)

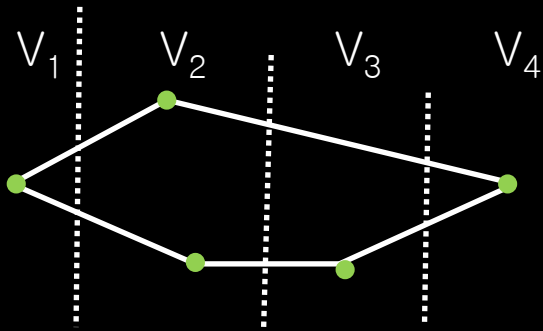
pick an embedding φ_i uniformly at random from E

$X \leftarrow X \cdot X_i$

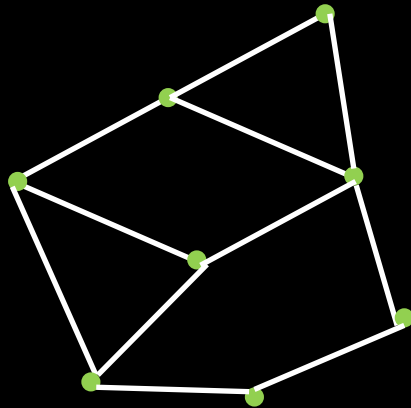
EndFor

Return $(\varphi_l, 1/X)$

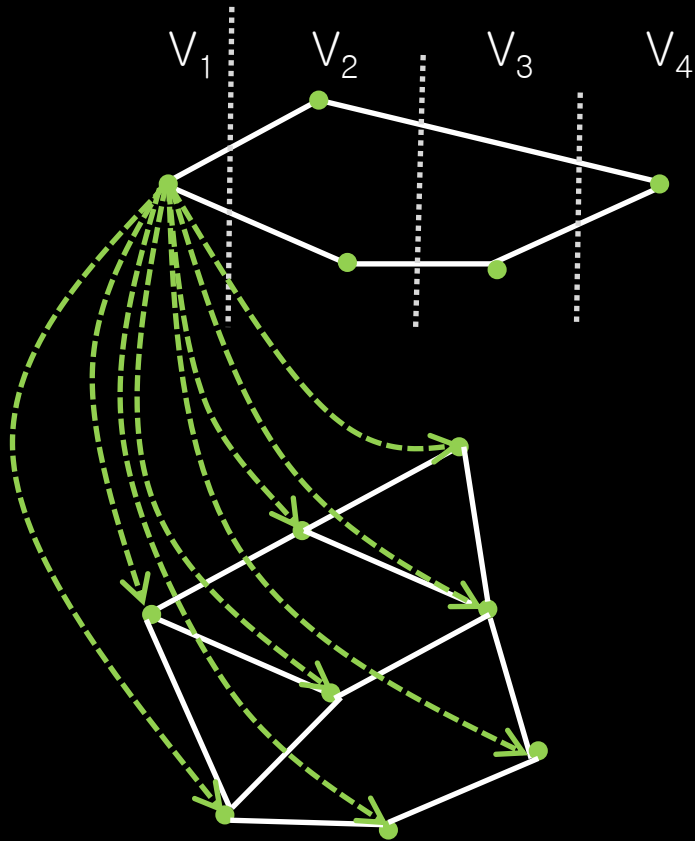
Fürer - Sample run (step 1.1)



- $X = 1; \varphi_0 = \{\}$
- $i = 1$

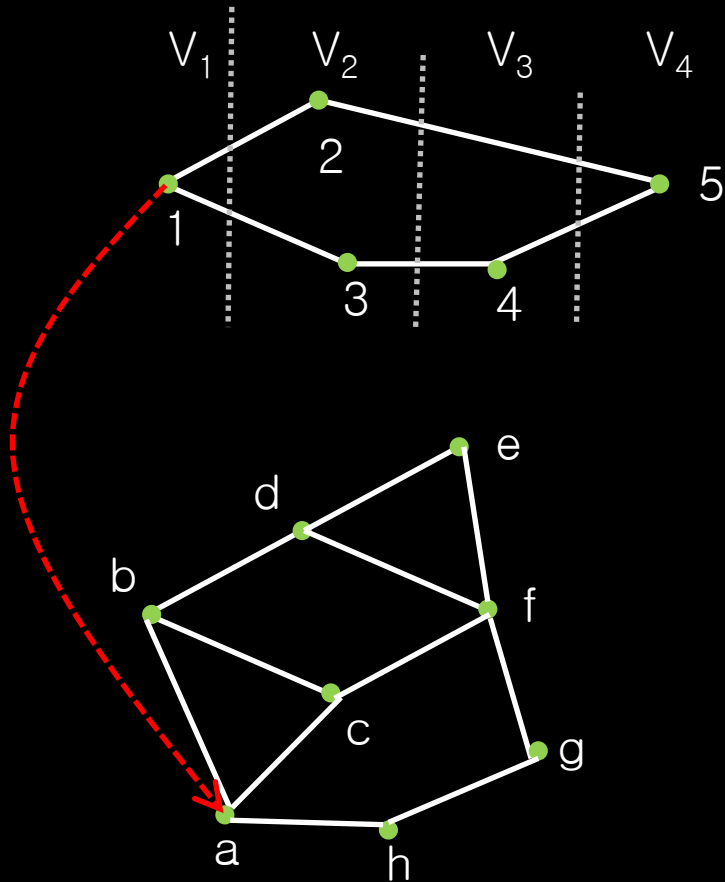


Fürer - Sample run (step 1.2)



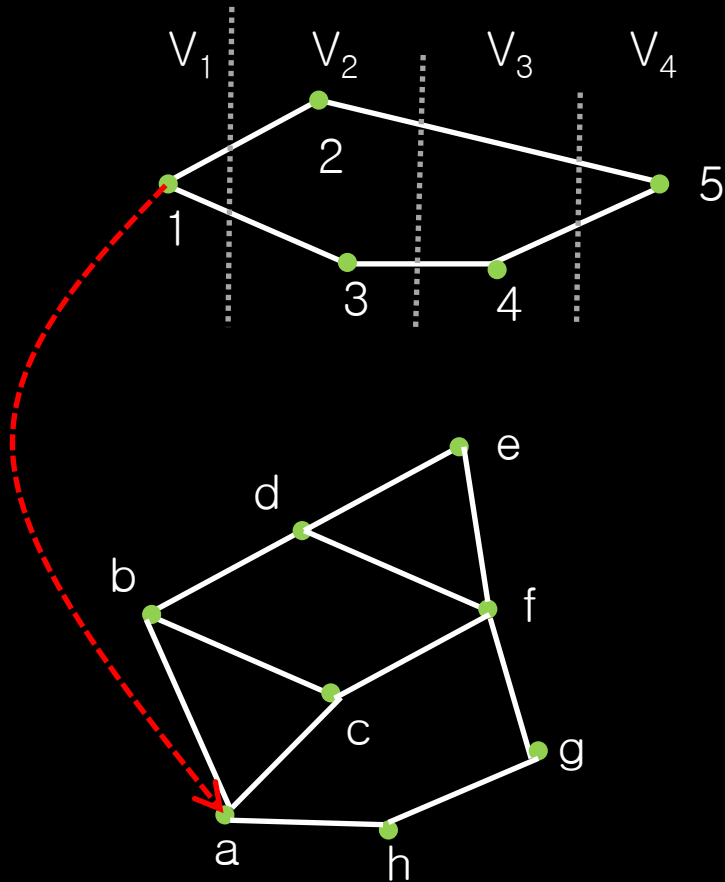
- $X = 1; \varphi_0 = \{\}$
- $i = 1$
- $X_1 = |E| = 8$

Fürer - Sample run (step 1.3)



- $i = 1$
- $X_1 = 8$
- $\varphi_1 = \{(1, a)\}$
- $X = 8$

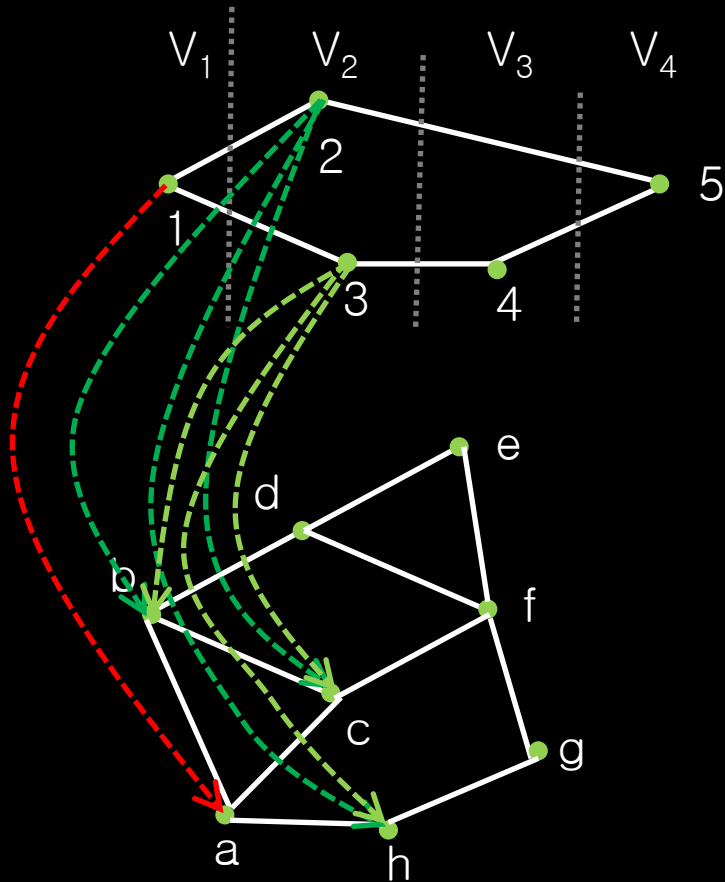
Fürer - Sample run (step 2.1)



- $i = 2$

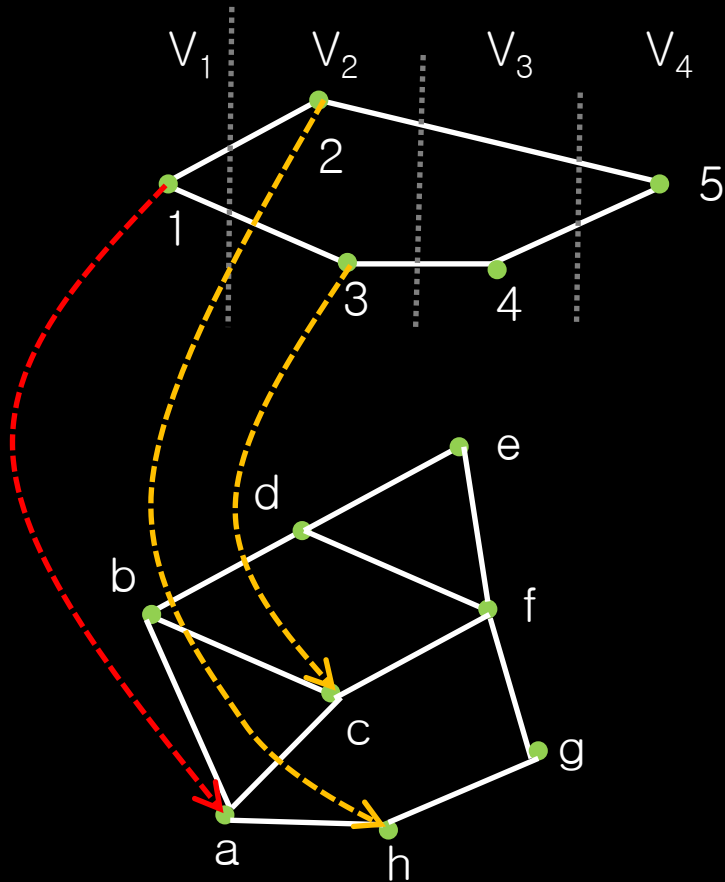
- $X = 8 ; \varphi_1 = \{(1, a)\};$

Fürer - Sample run (step 2.2)



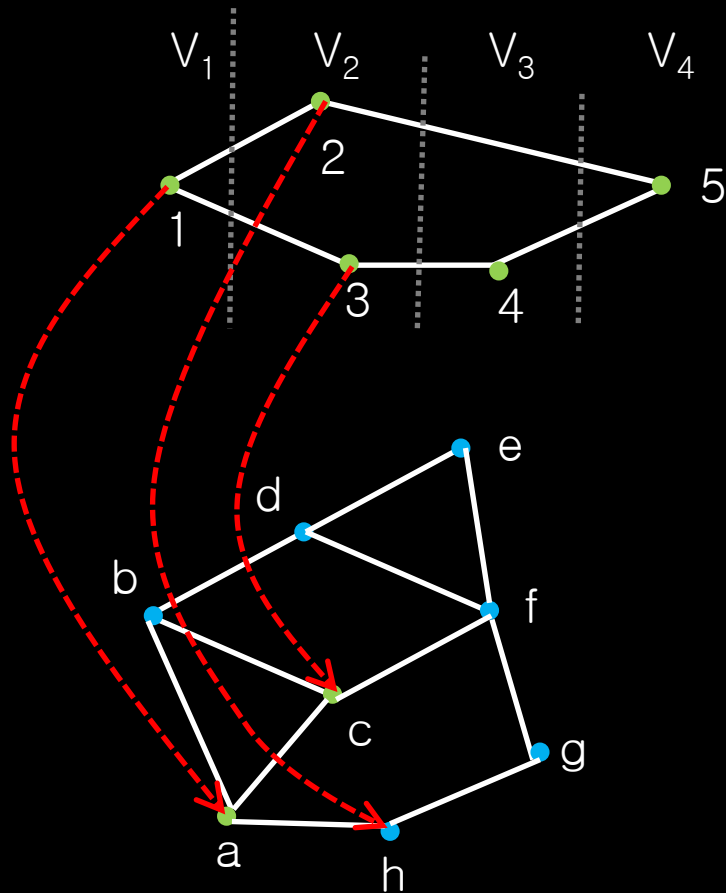
- $i = 2$
- $X = 8 ; \varphi_1 = \{(1, a)\};$
- $X_2 = |E| = 3.2 = 6$
- $X \leftarrow X.X_2 = 8 * 6 = 48$

Fürer - Sample run (step 2.3)



- $i = 2$
- $\varphi_1 = \{(1, a)\};$
- $X_2 = |E| = 3.2 = 6$
- $X = 48$
- $\varphi_2 = \{(1, a), (2, h), (3, c)\}$

Fürer - Sample run (step 3.1)

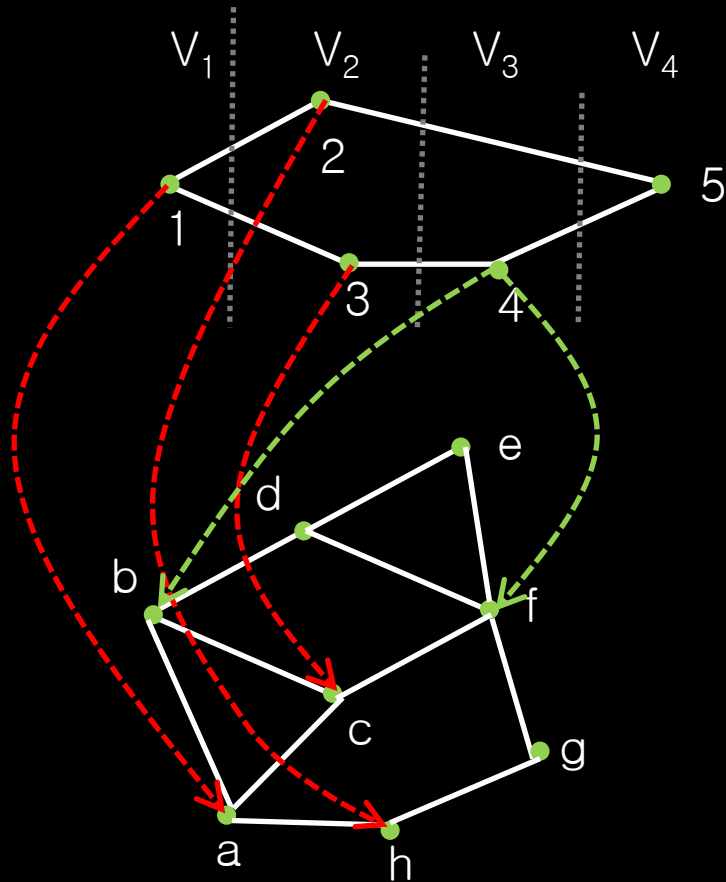


- $i = 3$

- $\varphi_2 = \{(1, a); (2, h), (3, c)\};$

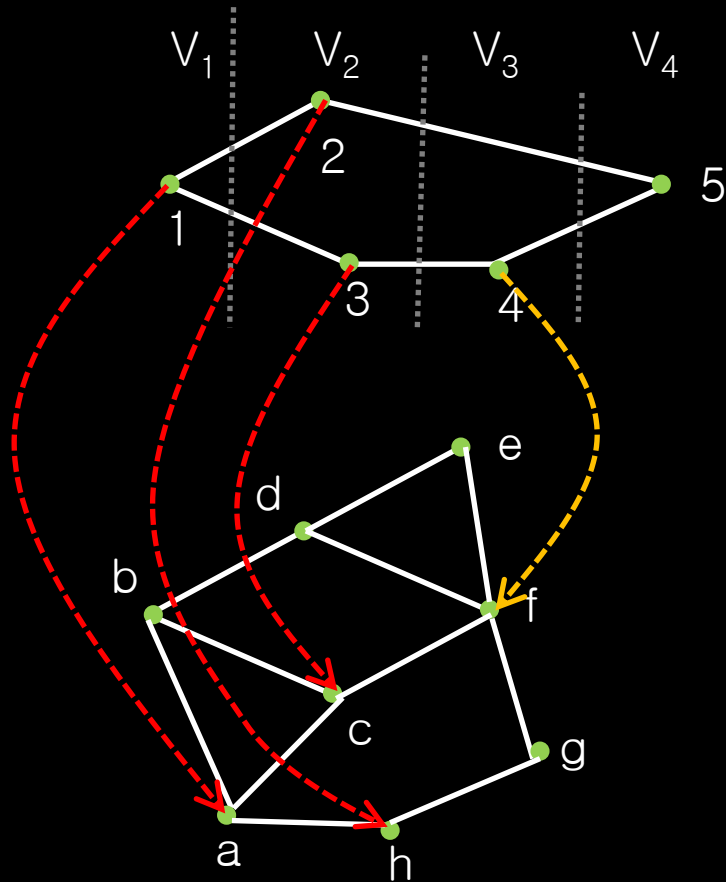
- $X = 48$

Fürer - Sample run (step 3.2)



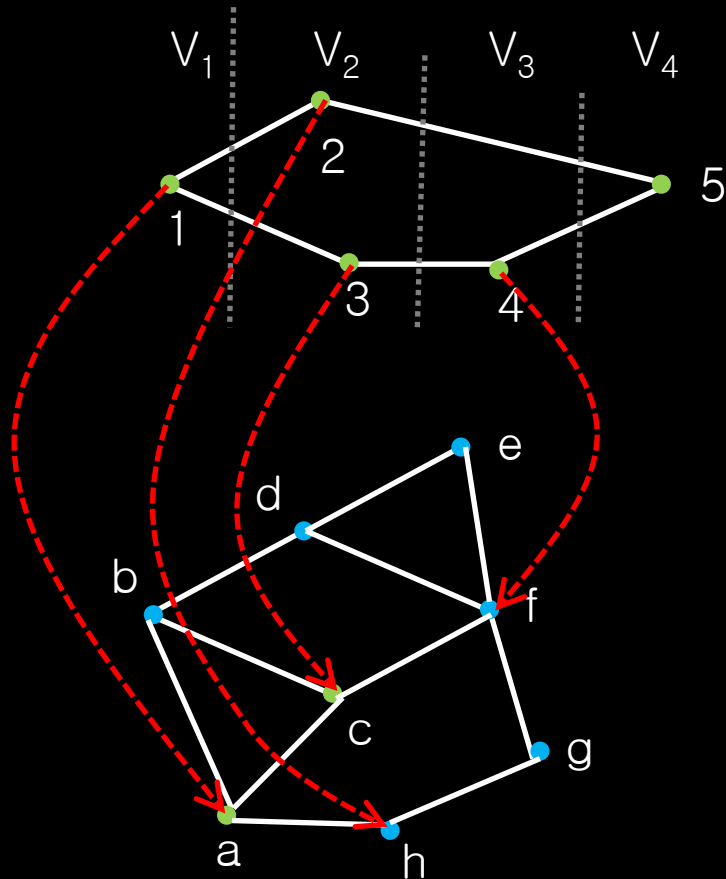
- $i = 3$
- $\varphi_2 = \{(1, a); (2, h), (3, c)\};$
- $X = 48$
- $X_3 = |E| = 2$
- $X = X \cdot X_3 = 48 * 2 = 96$

Fürer - Sample run (step 3.3)



- $i = 3$
- $\varphi_2 = \{(1, a); (2, h), (3, c)\};$
- $X = 48$
- $X_3 = 2; X = 96$
- $\varphi_3 =$
 $\{(1, a), (2, h), (3, c), (4, f)\}$

Fürer - Sample run (step 4.1)

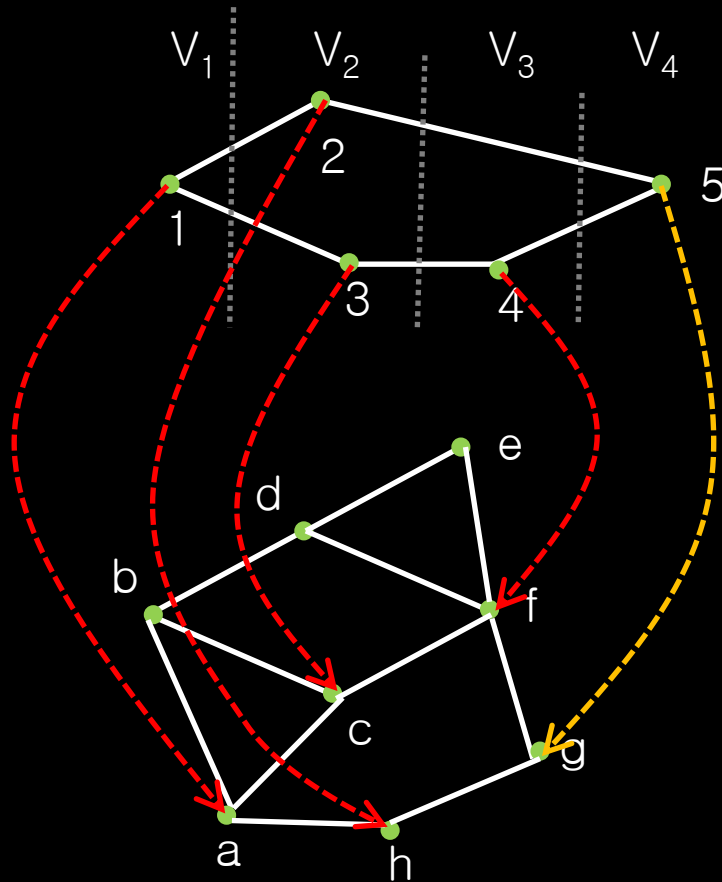


- $i = 4$

- $\varphi_3 = \{(1, a); (2, h), (3, c), (4, f)\};$

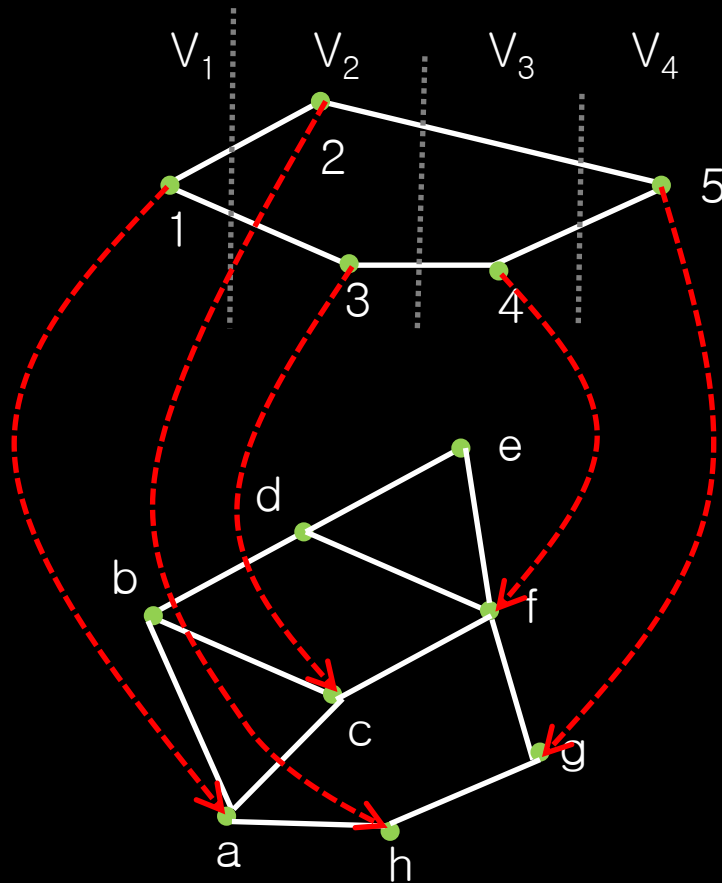
- $X = 96$

Fürer - Sample run (step 4.2)



- $i = 4$
- $\varphi_3 = \{(1, a); (2, h), (3, c), (4, f)\};$
- $X = 96$
- $X_4 = 1$
- $\varphi_4 = \{(1, a), (2, h), (3, c), (4, f), (5, g)\}$

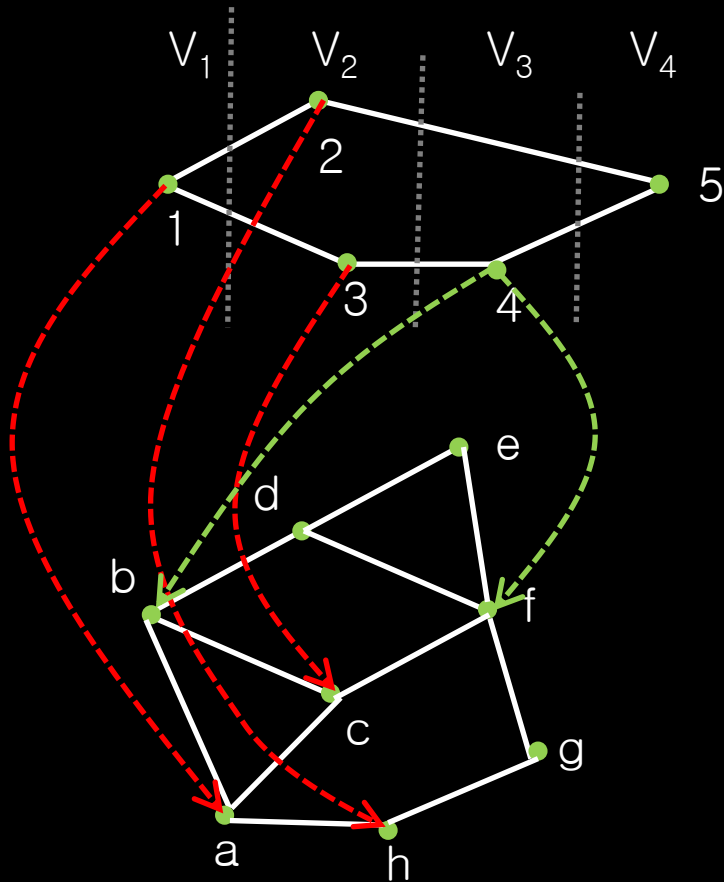
Fürer - Sample run - a solution



■ $X=96;$

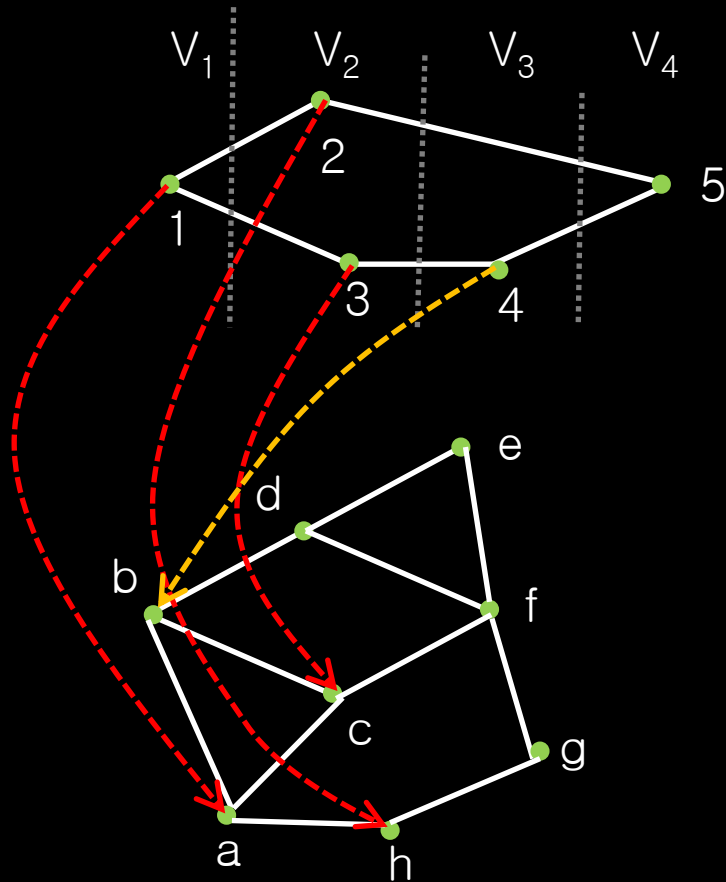
■ $\varphi=\{(1,a),(2,h),(3,c),(4,f),(5,g)\};$

Fürer - Sample run bis



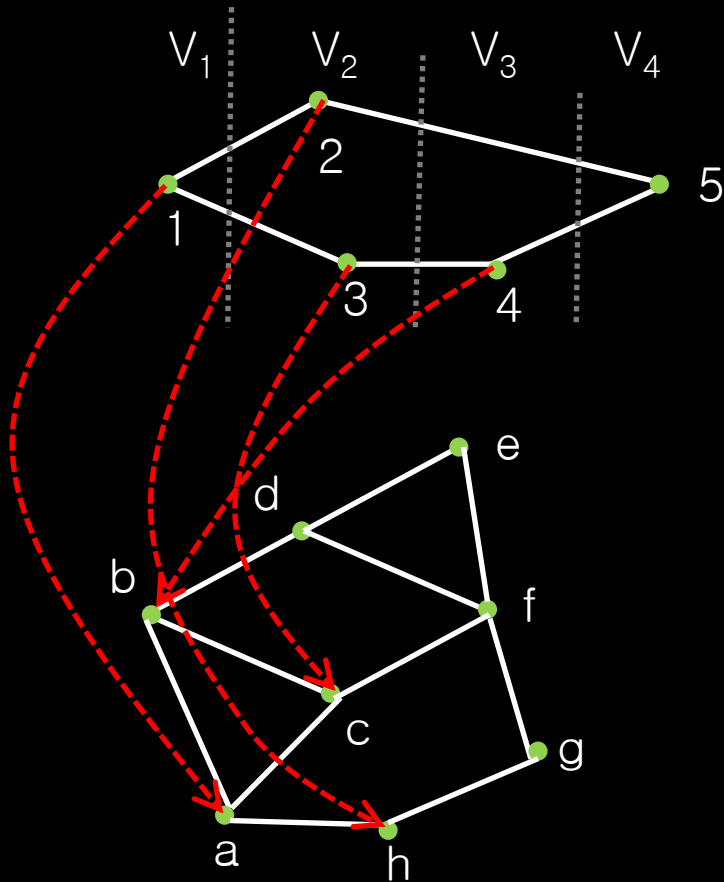
- $i = 3$
- $\varphi_2 = \{(1, a); (2, h), (3, c)\};$
- $X_3 = |E| = 2$
- $X = X \cdot X_3 = 48 * 2 = 96$

Fürer - Sample run bis



- $i = 3$
- $\varphi_2 = \{(1, a); (2, h), (3, c)\};$
- $X=96$
- $\varphi_3 = \{(1, a), (2, h), (3, c), (4, b)\}$

Fürer - Sample run bis



- $i = 3$
- $\varphi_3 = \{(1, a); (2, h), (3, c), (4, b)\};$
- $X = 96$
- $X_4 = 0 \rightarrow$ **no solution!**

Pattern matching

Fürer's theorem

- Important: When does it work?
- The algorithm is an FPRAS (for almost all G) if:
 - Partitioning of $V(H)$ is a “ordered bipartite decomposition”
 - $p(n)n^2 \rightarrow \infty$
 - $np^v/\Delta^4 \rightarrow \infty$
 - $E(H)^3n^{-4}p^{-2} \rightarrow 0$

Where

$$v = \max(2, \gamma), \quad \gamma = \max\{|E(F)|/(|V(F)|-2) \mid F \leq H, |V(F)| \geq 3\}$$

Pattern matching

Fürer: Simplified theorem

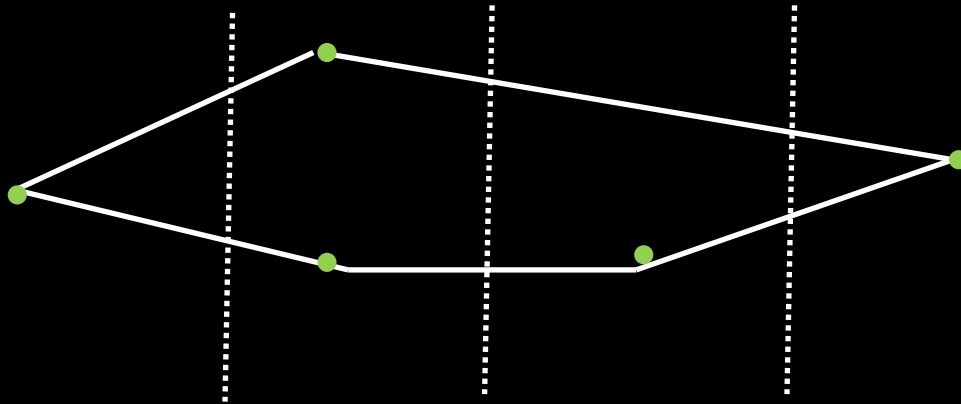
- It works if the count is not too small.
- Intuition:
 - The frequency of very rare events is hard to measure (the “interesting” part of the sample where we actually observe something, is smaller)
 - The worst case is where one doesn’t know whether $|Emb(H, G)| = 0$ or $|Emb(H, G)| = 1$
- If we run the algorithm and we find some embeddings, our estimate will be rather accurate.
- If no “ordered bipartite decomposition”, then we can still run this algorithm, but no FPRAS guarantee.

Pattern matching; Fürer

Ordered bipartite decomposition

- An **ordered bipartite decomposition** of H is a partition $\{V_1, V_2, \dots, V_l\}$ of $V(H)$ such that
 - Each $V_i (i=1..l)$ is an independent set in H
 - $\forall i, v \in V_i \exists j$ such that $N_H(v) \subseteq \bigcup_{k < i} V_k \cup V_j$
- So if a neighbor of a vertex $v \in V_i$ is in V_j with $j > i$, then no neighbors of v must be in $V_{j'}$ with $j' > i$ and $j \neq j'$.

Fürer - Graphs with a decomposition



- Cycles longer than 3 (see above running example)
- Bounded degree outerplanar graphs without triangle
- Trees
- Grids
- ...
- Not: triangles (but separate proof), ...

Pattern matching

Fixed parameter tractability


Pattern matching

Avoiding worst-case complexity

- Network

- ✗ No definite structure we can rely on
- ✗ Approximate matching may help but
 - Subgraph isomorphism is hard to approximate
- ✓ Structural properties of patterns
 - Triangles and other small patterns **DONE**
 - Trees
- ✓ Statistical properties of network
 - Random graphs **DONE**

Fixed
parameter
tractability



Pattern matching

Fixed parameter tractability

- Classic complexity classes:
 - Input size: n
 - P : Polynomial time $O(n^d)$ for some d
 - Are all problems not in P are hard: not $O(n^d)$ for some d ? No, some may still be easier than others
- Fixed parameter tractability:
 - Input has two parts of sizes: n and k
 - For k fixed, the problem is tractable: $O(n^d f(k))$
 - May be acceptable if k is small

Pattern matching

Fixed parameter tractability

- Fixed parameter tractability:
 - Input has two parts of sizes: n and k
 - For k fixed, the problem is tractable: $O(n^d f(k))$
 - May be acceptable if k is small
- Pattern matching:
 - Network: size n
 - Pattern: size k
 - k is usually small, n may be large.
- So fixed parameter tractability is suitable for pattern matching!

Pattern matching

Fixed parameter tractability

- Matching trees in network in
$$O(mk2^k) = O^*(2^k)$$
 - ▣ $k = V(P)$: pattern size
 - ▣ $m = E(D)$: network size
- Randomized algorithm¹ which works well for practical pattern mining²

¹ R. Williams, Inf Proc Lett 2009

² A. Kibriya & J.Ramon, DMKD 2013

Pattern matching summary

- Basic operation for mining and learning
- Networks have no hard structure we can rely on
 - ✗ Approximate matching may help but
 - Subgraph isomorphism is hard to approximate
 - ✓ Structural properties of patterns
 - Triangles and other small patterns
 - Trees
 - ✓ Statistical properties of network
 - Random graphs

Pattern matching – open problems

- Many matching problems for which fixed parameter approach could help
 - More complex queries
- Implicit relations
- Big data (not in RAM)
 - combine with indexing
 - reduce passes over data (or samples)
 - distributed approaches, ...

Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
 - Introduction
 - Pattern matching
 - **Frequency**
 - Additional remarks
- Learning

Frequency / support¹ measures

The problem

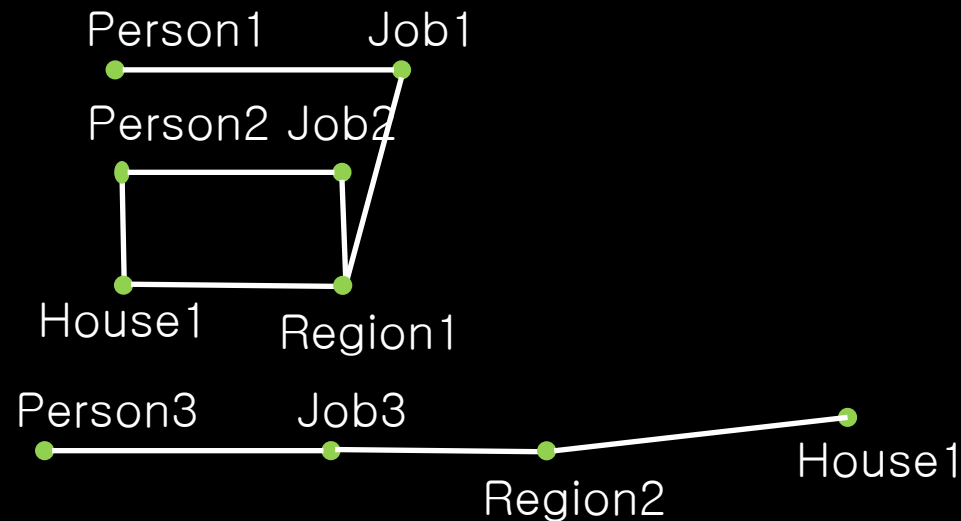
- How **frequent** is pattern P in network D ?
- Why assign a “frequency” to a pattern?
 - Popular criterion to measure relevance of pattern
 - E.g. 40% of respondents liked both movie m_1 and m_2 .
 - Way to represent association rules
 - E.g. Of all respondents liking movie m_1 and m_2 , 50% also liked m_3 (i.e. $40\% * 50\% = 20\%$)
 - Measure of statistical power
 - E.g. We rolled the dice 100 times and observed 40 times a 6. It must be biased.

¹ both “frequency” and “support” are used, often interchangeably

Support measures

What do you want?

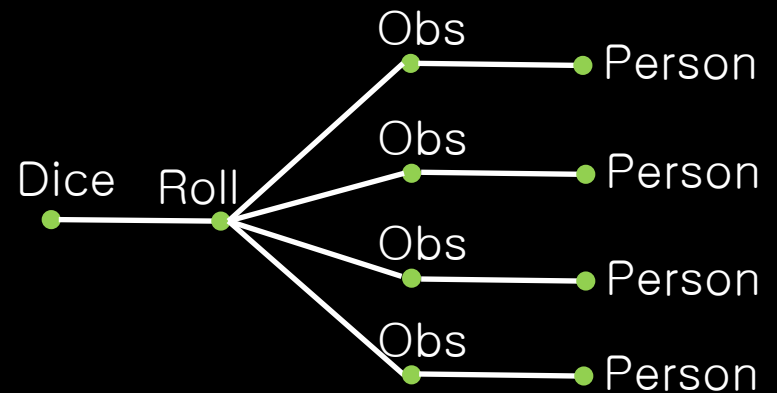
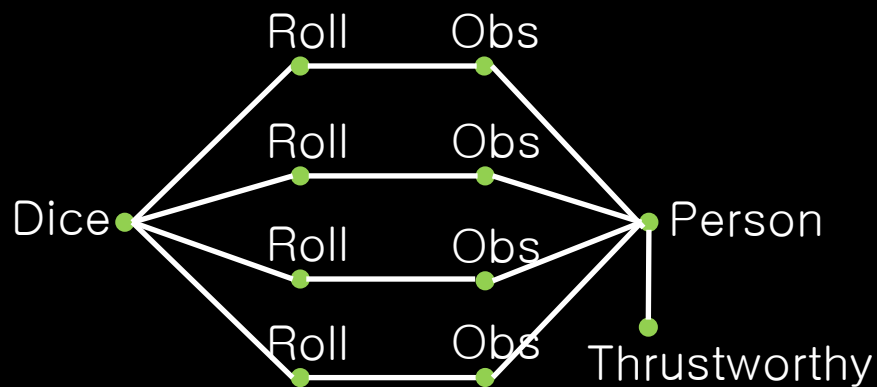
- Counting objects
 - X% of people own a house in the same region where they work.
 - network unimportant, just count people



Support measures

What do you want?

- Counting objects
- Performing statistics
 - We rolled the dice 100 times and observed 40 times a 6. It is biased!
 - We rolled the dice. 100 people observed a 6. Would it be biased?



Support measures

What do you want?

- Counting objects
- Performing statistics
- Association rules
 - If X has a friend Y such that Y smokes, then with probability $a\%$, X smokes too
 - If X and Y are friends and Y smokes, then with probability $b\%$, X smokes too
 - Different quantor/aggregator placement, probably $a \neq b$

Support measures

What do you want?

- What do you want?
 - Counting objects
 - Performing statistics
 - Association rules
 - ...
- If you know what you want, you're closer to knowing what to do. No measure is good in all cases.

Support measures

Overview

- Embedding-based
 - Embedding count
 - Image count
- Key-based
 - Key image count
 - Min-image
- Overlap-based
 - Maximum independent set
 - Minimum clique partition
 - Intermediate measures and relaxations

Support measures

Embedding-based

- Embedding-count: $|Emb(P, D)|$
- Image-count: $|Img(P, D)|$
 $|Emb(P, D)| = |Aut(P)| \cdot |Img(P, D)|$
- E.g.: *Among all triples (X, Y, Z) such that X and Y are friends and Z is a family member of X , the fraction of triples where Y knows Z is $a\%$.*
- Embeddings may be concentrated in small part of network, e.g. large family where everyone is friend with each other.
- Not anti-monotonic, no pattern-mining pruning

Support measures

Key-based

- Decide before the start of data mining what is the type of object of interest (“the primary key”)
- E.g. *We are interested in ‘friends’ relationships*
 - *a% of ‘friends’ relations are between colleagues.*
 - *b% of friends have the same mother tongue,*
 - *c% of friend pairs (X,Y) have at least one common friend Z,*
 - *...*

Support measures

Key-based

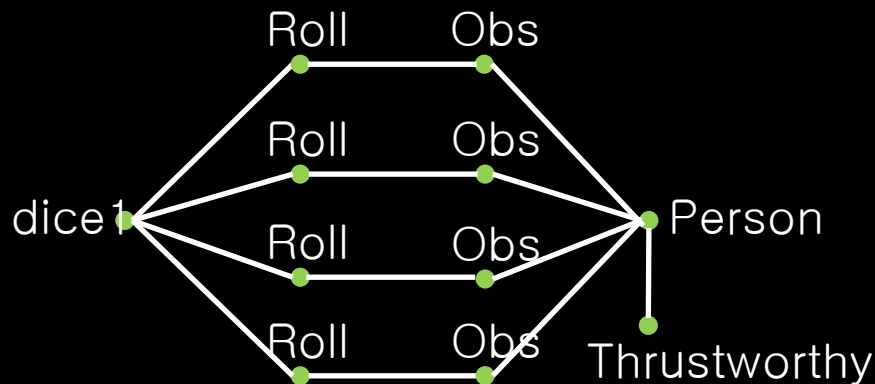
- Decide before the start on the “key”.
- The “key” is a common subpattern of all patterns considered.
- There is a fixed, finite set of objects (all images/embeddings of the “key”), the network relations are not considered.
- Easy to count
- Anti-monotonic (good for pattern mining)
- Not all statistics are valid (e.g. the dice example)

Support measures

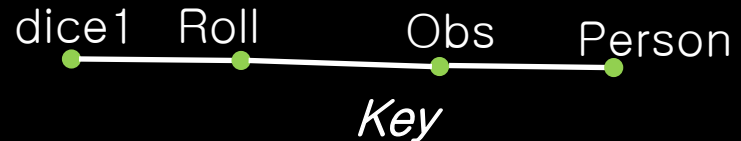
Key-based: dice example

- Key = dice observation
- Performing statistics:

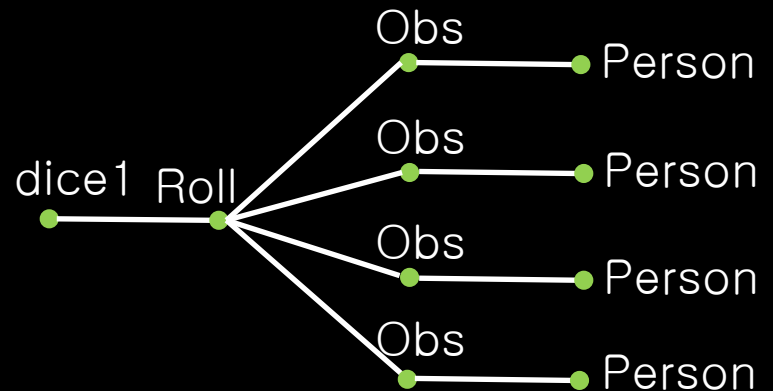
- We rolled the dice 100 times and observed 40 times a 6. It is biased!



4 images of **Key**
each showing a 6



- We rolled the dice. 100 people observed a 6. Would it be biased?



4 images of **Key**
each showing a 6

Support measures

Min-image¹

- $\text{minImage}(P, D) = \min_{v \in V(P)} |\{\pi(w) \mid \pi \in \text{Emb}(P, D)\}|$
- Allows for choosing each vertex as (singleton) key, giving a lower bound for each vertex-key-based frequency
- Anti-monotonic:
 $P \leq Q \Rightarrow \text{minImage}(P, D) \geq \text{minImage}(Q, D)$

¹ Bringmann & Nijssen. PAKDD 2008

Support measures

Overlap-based: model dependence

- The easiest way to perform statistics is to have **independent** observations.
- How do we get as much independent observations as possible out of a network?
- Model the independences in “overlap graph”.
- Caution: selecting independent observations is not necessarily a sample from the original distribution!

Support measures

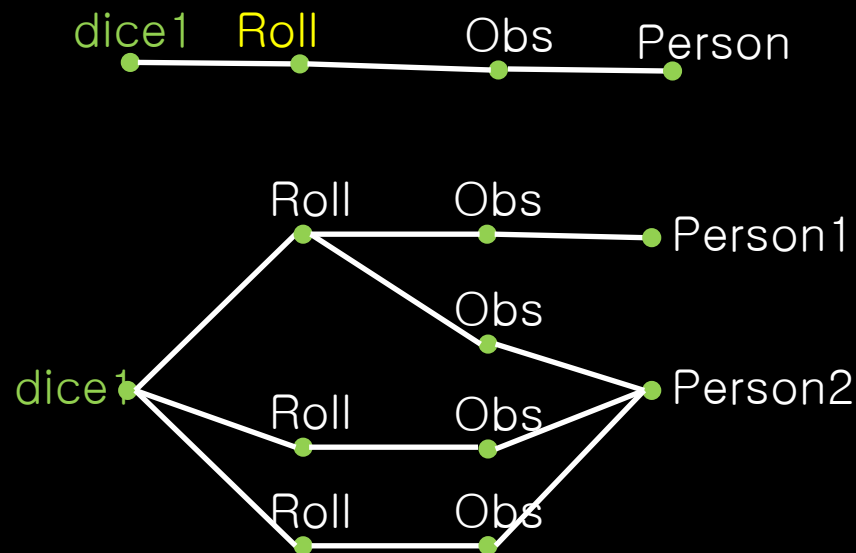
Overlap-based: Overlap graph

- Overlap graph G_P^D :
 - $V(G_P^D) = \text{Img}(P, D)$
 - $(g_1, g_2) \in E(G_P^D)$ if images g_1 and g_2 overlap
- What is overlap?
 - Two occurrences of a pattern overlap if we can't consider them independent in the context of the statistics we are doing
 - Vertex-overlap: $(g_1, g_2) \in E(G_P^D) \Leftrightarrow V(g_1) \cap V(g_2) \neq \emptyset$
 - Edge-overlap: $(g_1, g_2) \in E(G_P^D) \Leftrightarrow E(g_1) \cap E(g_2) \neq \emptyset$
 - Other options, e.g. Harmful overlap¹

Support measures

Overlap-based: What is overlap?

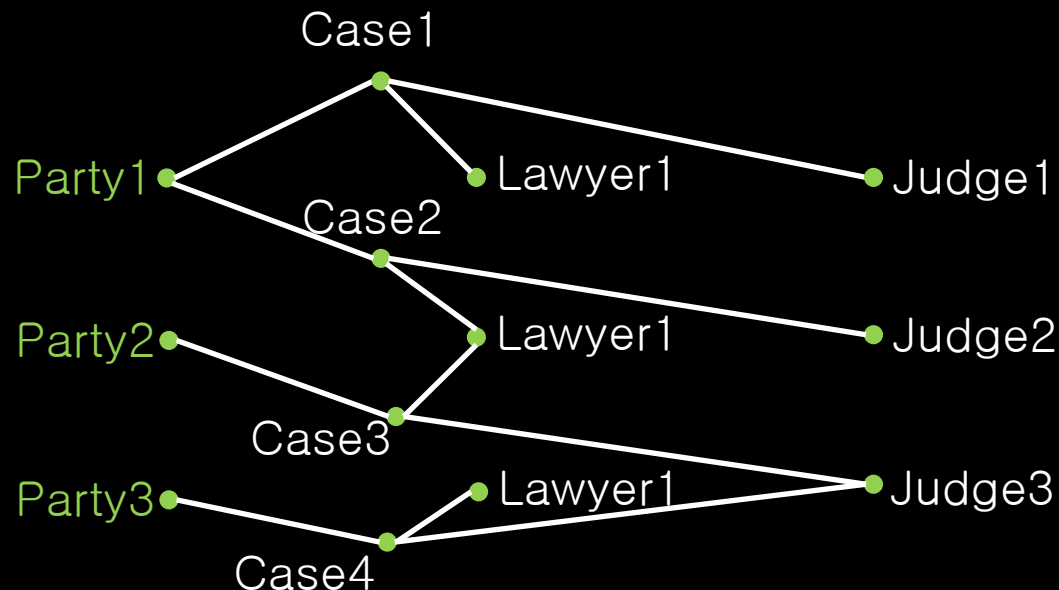
- Dice example:
 - Overlap if the *Roll* is the same.
 - We are interested in *dice1*, so naturally all embeddings will contain *dice1*.



Support measures

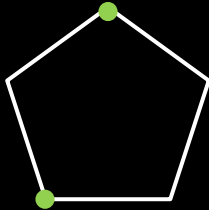
Overlap-based: Court example

- Vertices: case, judges, (prodeo) lawyers, party
- Edges between case-judge, case-lawyer, case-party

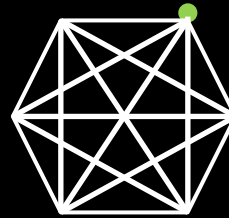


Independent set

- W is an **independent set** of H iff
 - $W \subseteq V(H)$
 - There are no $v, w \in W$ such that $(v, w) \in E(H)$



Independent set of size 2

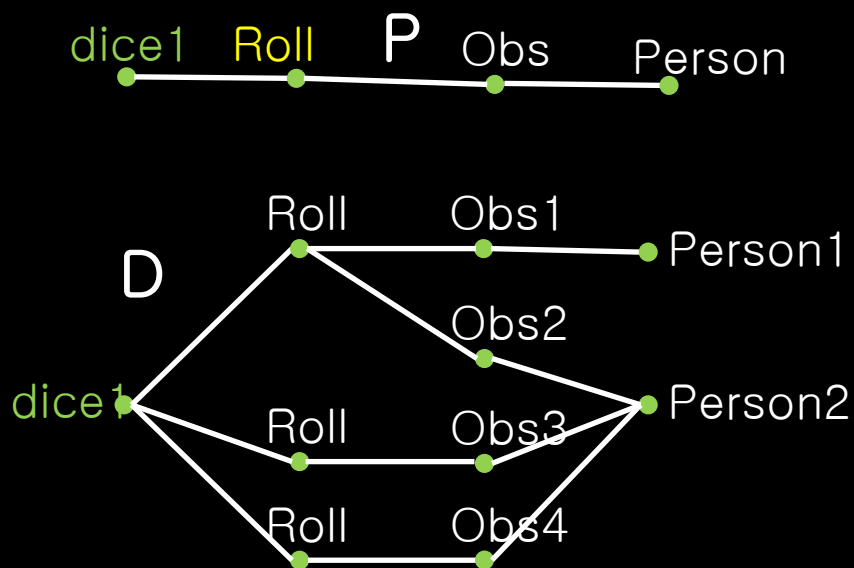


Independent set of size 1

Support measures

Overlap-based: MIS measure

- The size of a Maximum Independent Set (MIS) of the overlap graph G_P^D of pattern P in network D : $MIS(P, D) = MIS(G_P^D)$



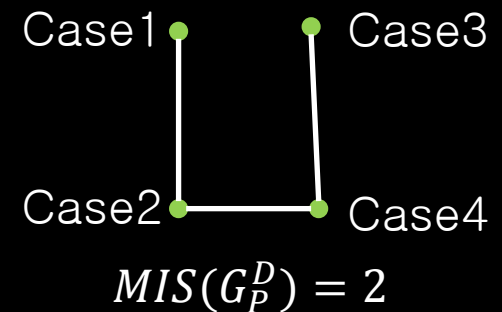
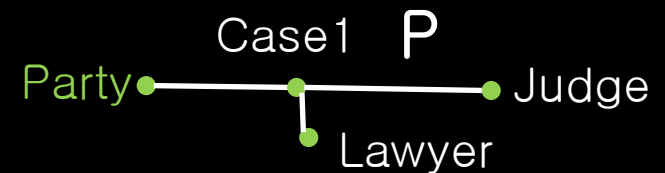
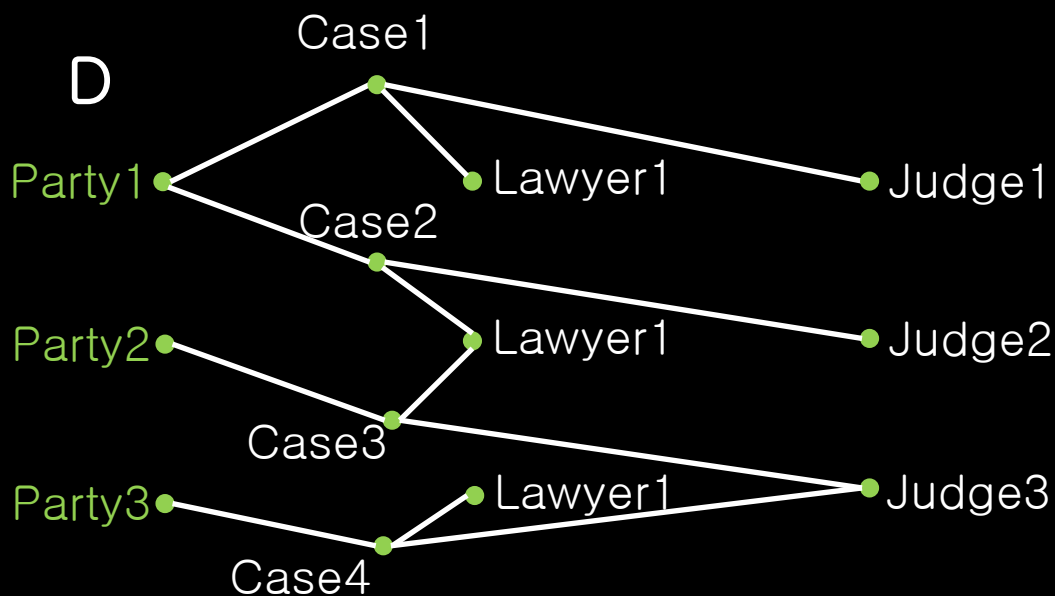
$$MIS(G_P^D) = 3$$

- $(\text{dice1}, \text{obs}=6, \text{Person1})$
- $(\text{dice1}, \text{obs}=6, \text{Person2})$
- $(\text{dice1}, \text{obs}=6, \text{Person2})$
- $(\text{dice1}, \text{obs}=3, \text{Person2})$

Support measures

Overlap-based: MIS measure

- The size of a Maximum Independent Set (MIS) of the overlap graph G_P^D of pattern P in network D : $MIS(P, D) = MIS(G_P^D)$



Support measures

Overlap-based: more measures

- Maximum independent set idea:
 - 😊 allows to measure overlap and extract independent observations
 - 😊 Anti-monotonic
 - 😞 NP-hard to compute
 - 😞 Possibly ignores too much information
- Does the overlap graph allow for other measures?

Support measures

Overlap-based: more measures

- Requirements for feasible measure:
 - Anti-monotonic in pattern:
 - $p \leq P \Rightarrow f(p, D) \geq f(P, D)$
 - Monotonic in network:
 - $D \leq D' \Rightarrow f(P, D) \leq f(P, D')$
 - Normalized
 - If there are n independent (non-overlapping) observations (overlap graph = n isolated vertices), then support is n .

Support measures

Overlap-based: more measures

- If f is a function on the overlap graph, and f is feasible measure, then:

$$MIS(P, D) \leq f(P, D) \leq MCP(P, D)$$

- where $MCP(G_P^D)$ is the minimum clique partition number of the overlap graph, another NP-hard to compute number.

Support measures

Overlap-based: more measures

- Fortunately, several efficiently computable functions are between MIS and MCP .
- Lovasz theta¹: ϑ
 - feasible measure²; computable with semidefinite program (SDP), which is still rather expensive
- MIS-relaxation³: s
 - Is a feasible measure³; computable with linear program (LP), hence efficiently.
- We have $MIS \leq \vartheta \leq s \leq MCP$

¹ D. Knuth, Electr. J. Combin 1994

² Calders et al. DMKD 2011

³ Wang & Ramon DMKD 2013

Support measures

Summary

| Measure | Anti-Monotonic? | Statistics? | Efficient? |
|----------------------|-----------------|-------------|------------|
| Embedding count | ✗ | ✗ | ✓ |
| Image count | ✗ | ✗ | ✓ |
| key image count | ✓ | ✗ | ✓ |
| min-image | ✓ | ✗ | ✓ |
| Overlap-MIS | ✓ | ✓ | ✗ |
| Overlap-MCP | ✓ | ? | ✗ |
| Overlap- ϑ | ✓ | ✓ | ? |
| Overlap-s | ✓ | ✓ | ✓ |

Frequency – open problems

- Combine pattern matching and frequency
- Exploit network structure to increase speed (methods up to now don't)

Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
 - Introduction
 - Pattern matching
 - Frequency
 - **Additional remarks**
- Learning

Expected number of embeddings

- Let $D \sim G(n, p)$, then

$$|Emb(P, D)| = n^{|V(P)|} p^{|E(P)|} (1 - p)^{\frac{n(n-1)}{2} - |E(P)|}$$

- For small p :

$$|Emb(P, D)| = n^{|V(P)|} p^{|E(P)|}$$

- For trees:

$$|Emb(P, D)| = (np)^{|V(P)|} / p$$

- Often D is connected and $np > 1$

Expected number of embeddings

- # of expected embeddings grows with pattern size for sparse patterns
- Denser patterns may be easier to interpret
- Also patterns less frequent than expected may be of interest.
 - ▣ This also happens with itemsets: items may be correlated, uncorrelated or anti-correlated

Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
- Learning
 - Introduction
 - Learning from non-independent examples
 - Temporal models

Learning - introduction

- Popular learning ideas:
 - connected vertices have similar target value
 - correlation between features and target value
 - more classic feature-to-target supervised learning
 - “Dual space” idea: one feature per vertex u , is 1 for vertices connected to that vertex u (else 0).
 - If individuals are important (but not many features are known)

Learning – introduction

Similar to your neighbor

- Semi-supervised learning
 - E.g.: try to minimize the number of edges with on both sides different labels
 - E.g. target values tend to average of neighbors.
 - ...
- Manifold embedding:
 - Assign all vertices a coordinate such that connected vertices are close together (and not-connected vertices are far apart)

Learning – Introduction

From feature to target value

- Learning tasks¹:
 - Vertices / edges / ...
 - (existence)prediction / labeling / weighting / feature construction / ...

¹ Rossi et al. “Transforming graph data for statistical relational learning”, JAIR 2012

vertex/edge structure/ feature prediction example

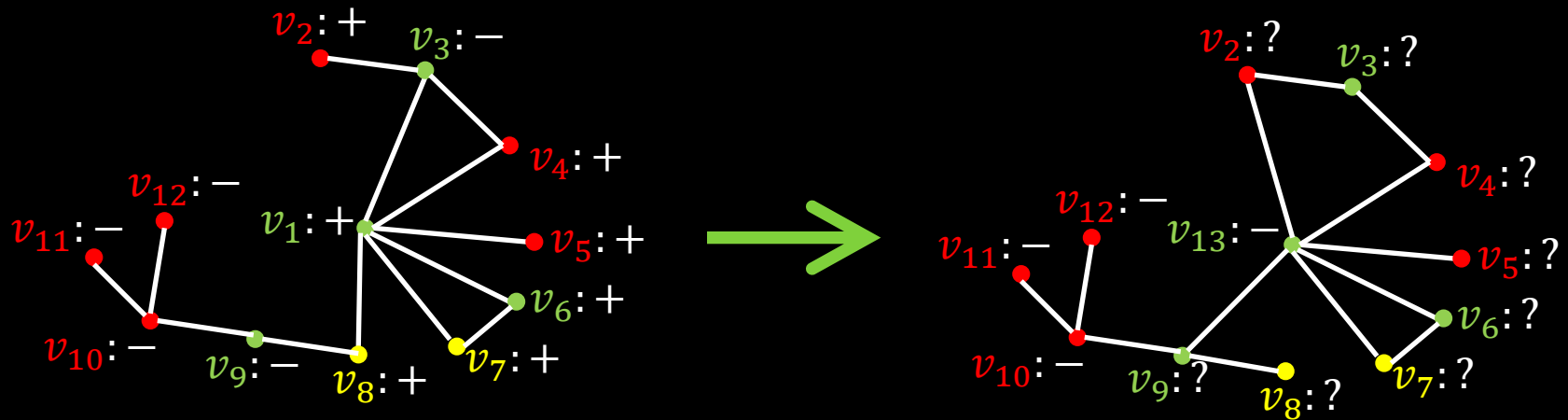
- Supervised learning
 - Input:
 - vertex/edge to predict
 - Neighborhood
 - Output:
 - Label or existence
- But how about the classic i.i.d. assumptions?

Learning – Prediction in a fixed network

- Most common setting:
 - Fixed network D
 - training and test vertices (edges) in D
- But what if the network changes (e.g. an influential node is added/deleted)?
 - Causal patterns remain the same
 - Correlation patterns may change significantly

¹ Rossi et al. “Transforming graph data for statistical relational learning”, JAIR 2012

Learning – Can't distinguish individual and its features



What is the rule?

Does everyone follow the class of its neighbors?

Is v_1 very influential?

Is class $+$ if there are green neighbors?

Learning – network-specific challenges

- Suppose the same rules stay true, but new nodes/edge (same distribution). Distinction viral/features may matter
- Movie database:
 - Persons from a fixed distribution
 - Movies from a fixed distribution
 - Persons watch movies
 - A few new persons and movies are added

Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
- Learning
 - Introduction
 - Learning from non-independent examples
 - Temporal models

Learning from dependent examples

- Members of a network are dependent
- Typical assumptions of learning algorithms don't hold, in particular that
 - Examples are independently and identically drawn (i.i.d.)

Movie rating example

- Movie rating
 - Obj: Movie (genre, duration, actor popularity)
 - Obj: Person (age, gender, ...)
 - Obj: Screening (location, time, ...)
 - Target: Rating
- Several ratings per person / movie / cinema

Lawsuit example

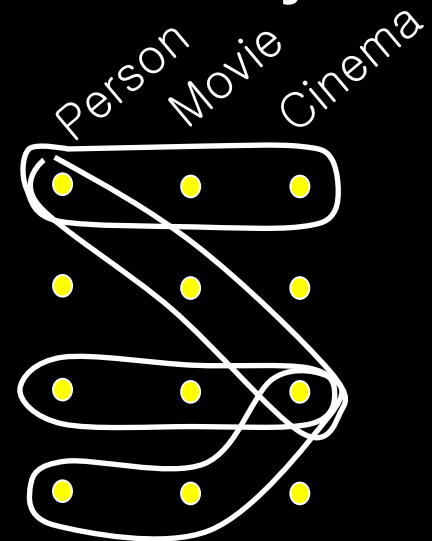
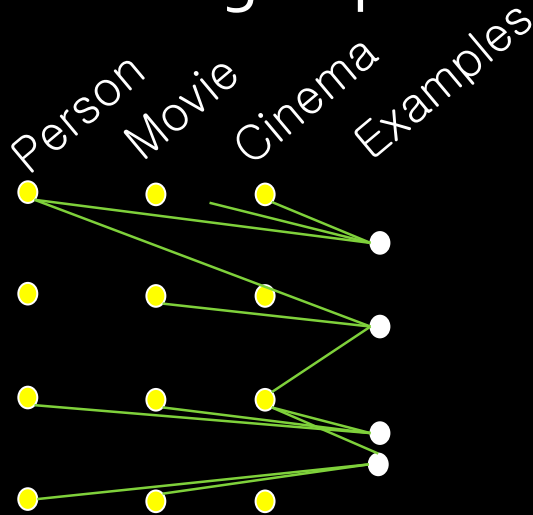
- Lawsuits:
 - Obj: Person
 - Obj: Lawyer
 - Obj: Judge
 - Example: case
 - Target: outcome
- Judges handle several cases, persons may be involved in several cases

Learning from pattern features

- Each example is an embedding of a pattern
 - MovieRating: (Movie, Person, Cinema, Rating)
 - Lawsuit: (Case, Person, Judge, Outcome)
- Examples overlap:
 - See also “support measures”
 - We call them **networked examples**

Representing networked examples

- Several alternative equivalent representations:
 - Every example is represented with a vertex connected to the participating objects
 - Every example is represented with a hyperedge, containing all participating + all relevant objects.



Learning from networked / dependent examples

- Tasks:
 1. Elementary statistics, confidence intervals, hypothesis testing, ...
 2. Learning, generalization guarantees
- Models:
 - a. bounding covariance between dependent examples
 - b. modeling how examples are dependent
- Combinations: 1a and 2b

Bounding covariance of examples and hypothesis testing

- (Wang, Neville, Gallagher, Eliassi-Rad, ECML/PKDD-2011) :
 - vertices are examples
 - edges indicate a bounded covariance
- safe correction for statistical significance tests
- safe upper bound for variance on sums etc.
- Upper bound for variance can be an important tool in proving generalization guarantees.

Bounding covariance of examples

- **n independent** random variables $\{X_i\}_{i=1}^n$ with variance σ^2 : Variance on $\sum_{i=1}^n X_i$ is $n\sigma^2$
- **n identical** random variables $\{X_i\}_{i=1}^n$ with variance σ^2 : Variance on $\sum_{i=1}^n X_i$ is $n^2\sigma^2$

Bounding covariance of examples

- **n networked examples**

- No edge between X_i and X_j = independent :

$$E[(X_i - E[X_i])(X_j - E[X_j])] = 0$$

- Edge $(X_i, X_j) \in E(D)$ between X_i and X_j = bounded covariance $E[(X_i - E[X_i])(X_j - E[X_j])] \leq \gamma$

- Variance on $\sum_{i=1}^n X_i$ is

$$\sum_{i,j} E[(X_i - E[X_i])(X_j - E[X_j])] \leq |E(D)|\gamma + n\sigma^2$$

Learning from networked / dependent examples

- Tasks:
 1. Elementary statistics, confidence intervals, hypothesis testing, ...
 2. Learning, generalization guarantees
- Models:
 - a. bounding covariance between dependent examples
 - b. modeling how examples are dependent
- Combinations: 1a and 2b

Variance, significance, effective sample size

- Effective sample size of a given set of networked examples is n iff it contains as much information (for the task at hand, e.g. learning or hypothesis testing) as a set of n i.i.d. examples¹.

¹ Slightly different conventions/definitions exist

Probably approximately correct (PAC) structure

- PAC: with probability $1 - \delta$ the loss is bounded by ϵ where

$$\delta = \exp\left(\frac{-m(S)\epsilon^2}{C_1 + C_2\epsilon}\right)$$

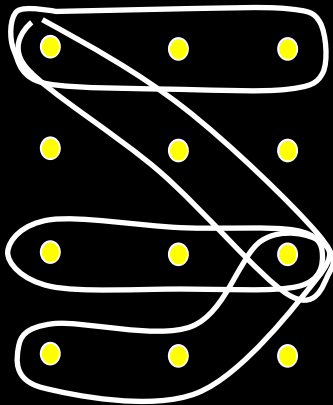
- with $m(S)$ the effective sample size of training set S . Higher $m(S)$ = better
- i.i.d. sample S , $m(S) = |S|$ = best possible

Independence assumptions

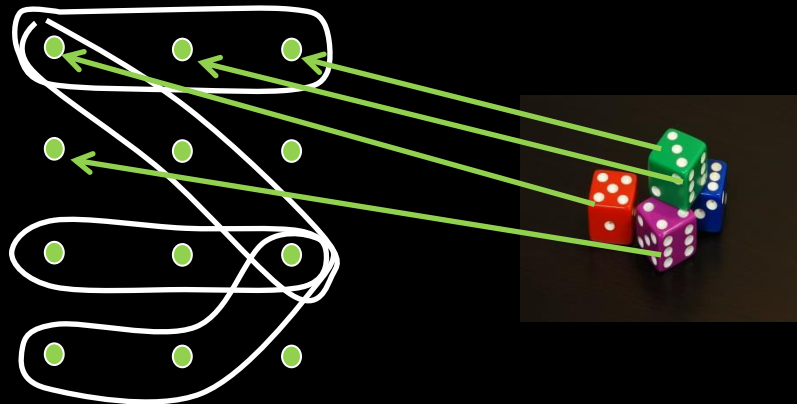
- Weaker form of i.i.d
- But not arbitrary
 - arbitrary \Rightarrow no bound possible

Independence assumptions: i.i.d. vertex features

- Edges are fixed.
- The features of every vertex are drawn i.i.d. (not even depending on the edges).



1. Choose edges
(possibly very
dependently)



2. Draw vertex
features (don't
look at edges)



Independence assumptions applied

YES

- Sneak preview
- Randomized trial: patients are assigned randomly to set of treatment params
- Cases are assigned randomly to judges

NO

- Select movie based on genre, or with friends
- Patients go to closeby hospital or to hospital recommended by their friends
- Judges handle cases connected to their existing cases

Training set measures

- Overlap (hyper)graph G
 - vertices are objects
 - (hyper)edges are examples
- Training set $S \subseteq E(G)$
- Measures $m(S)$ of training set $m(S)$:
$$\max(n_{EQW}, n_{IND}) \leq n_{MIS} \leq \vartheta \leq s \leq n_{MSC}$$

Approach 1: Equal-weight (EQW)

$$\max(n_{EQW}, n_{IND}) \leq n_{MIS} \leq \vartheta \leq s \leq n_{MSC}$$

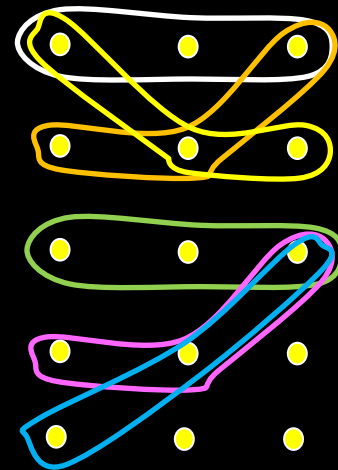
↑
all examples get same weight

(Janson 2004) & (Usunier 2005)

$$\delta \propto \exp\left(\frac{-n_{EQW}\epsilon^2}{C_1 + C_2\epsilon}\right)$$

With $n_{EQW} = \frac{|S|}{\chi^*(G)}$

and $\chi^*(G)$ fractional edge chromatic number



$$\chi^*(G) = 3 ; n_{EQW} = \frac{6}{3} = 2$$

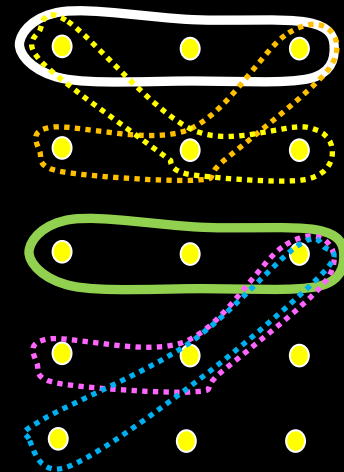
Approach 2: Independent set (IND)

$$\max(n_{EQW}, n_{IND}) \leq n_{MIS} \leq \vartheta \leq s \leq n_{MSC}$$

We find n_{IND} independent examples

$$\delta \propto \exp\left(\frac{-n_{IND}\epsilon^2}{C_1 + C_2\epsilon}\right)$$

With $n_{IND} = |S|$ (examples in S independent!)



$$n_{IND} = 2$$

Approach 3:

Maximum independent set

$$\max(n_{EQW}, n_{IND}) \leq n_{MIS} \leq \vartheta \leq s \leq n_{MSC}$$

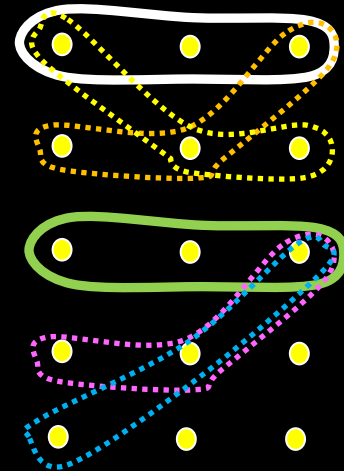
Maximum independent set of examples

$$n_{MIS} = |MIS(G)|$$

$|MIS(G)|$ hard to approximate!

⇒ no const lower bound for $\frac{n_{IND}}{n_{MIS}}$

For some G , $\frac{n_{EQW}}{n_{MIS}} = 2/|S|$



Minimum clique partition number

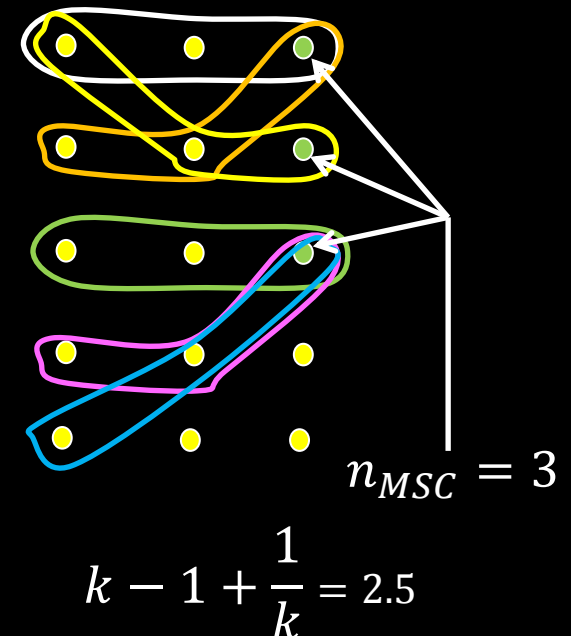
$$\max(n_{EQW}, n_{IND}) \leq n_{MIS} \leq \vartheta \leq s \leq n_{MSC}$$

Minimum set cover

$$n_{MSC} = |MSC(G)|$$

$$n_{MSC} \leq \left(k - 1 + \frac{1}{k}\right) n_{MIS}$$

$|MSC(G)|$ too is hard to compute

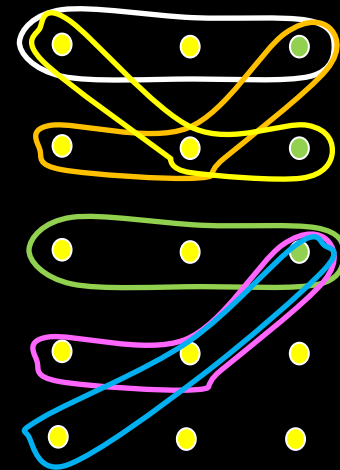


Approach 4: MIS-relaxation

$$\max(n_{EQW}, n_{IND}) \leq n_{MIS} \leq \vartheta \leq \mathbf{s} \leq n_{MSC}$$

↑
LP relaxation of MIS

- (Wang & Ramon, DMKD 2013)
- Graph pattern support measure
 - Anti-monotonic, Normalized
- Linear program \rightarrow efficient



$$s = 2.5$$

networked PAC

- PAC: $P(\text{Loss} \leq \epsilon) \geq 1 - \delta$ where

$$\delta = \exp\left(\frac{-s\epsilon^2}{C_1 + C_2\epsilon}\right)$$

- with s

$$s = \max\left(\sum_{i=1}^{|S|} w_i\right)$$

subject to

$$\forall v \in V(G) : \sum_{e_i: v \in e_i} w_i \leq 1$$

Networked PAC

- Influence of each factor is at most 1:

max s

$$s = w_1 + w_2 + w_3 + w_4 + w_5 + w_6$$

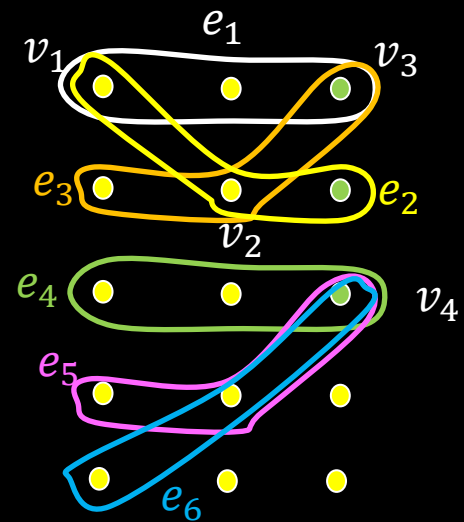
s.t.

$$v_1: w_1 + w_2 \leq 1$$

$$v_3: w_1 + w_3 \leq 1$$

$$v_2: w_2 + w_3 \leq 1$$

$$v_4: w_4 + w_5 + w_6 \leq 1$$



$$s = 2.5$$

Technical elaboration

- Chernoff bound for weighted sums:

For X_i ($i = 1..n$) independent random variables,

$E[X_i] = 0$; $|X_i| \leq a_i + M$; $X = \sum_i X_i$:

$$P\left(\sum_{i=1}^n X_i \geq n\epsilon\right) \leq \exp\left(\frac{-n\epsilon^2}{\text{Var}(X) + \sum_i a_i + M\epsilon/3}\right)$$

- Let $X_i = \xi(\{\phi(v)\}_{v \in e_i})$ and $\forall v : \sum_{e_i: v \in e_i} w_i \leq 1$,
then the above Chernoff inequality still holds

Learning from non-independent examples: Summary

- Networks modeling relations induce dependencies between objects and examples
- Modeling such dependencies is useful to
 - better understand the learning setting
 - get more statistical power from the data
 - Upper bound correlation between examples
 - Model common factors of examples

Learning from non-independent data: Open problems

- Can we formalize & structure models for learning?
- How to extract most statistical value from data?
- How to take intervention into account (very important for applications) ?

Contents

- Introduction
- Networks from different points of view
- Patterns & pattern mining
- Learning
 - Introduction
 - Learning from non-independent examples
 - Temporal models

Temporal models

- Temporal model = probability distribution mapping a network on a new network, i.e.
 $h: \mathcal{G} \times \mathcal{G} \rightarrow [0,1]$ s.t. $\forall D \in \mathcal{G}, \sum_{D' \in \mathcal{G}} h(D, D') = 1$
- Results in Markov chain
 - When starting from empty network: generative model
- Often one attempts to find a simple rule, when provided to all individuals of a group producing an interesting/real-life pattern
 - Communities, powerlaws, emerging structures, ...

Temporal models

- Global temporal models
 - How will communities evolve?
 - How will global network properties evolve?
 - Models aiming at emerging behavior
- Local temporal models
 - How will individual nodes/edge/local neighborhoods evolve?

Global temporal models

Examples

- How will communities evolve?
 - ▣ Research topics emerging and disappearing¹
 - ▣ Online groups emerging and disappearing²

1 Ferlez et al. ICDE 2008

2 Kairam, Wang & Leskovec WSDM 2012

Models for emerging or asymptotic patterns - Examples

- Theory: Barabasi-Albert: Preferential attachment
- Complex systems:
 - Economics (e.g. company fusions¹)
 - Between networks and physics: Groups of animals (formations of flying birds²)
- No systematic integration with DM/ML

¹ Garnett & Mollan, ECCS 2012

² Hemelrijk & Hildenbrandt, ECCS 2012

Local temporal models

- Local models:
 - Link prediction: what could exist may get known as existing soon.
 - Learn from temporal data
- Few combined local/global approaches
 - Emerging behavior shown by simulation (Complex systems)
 - Model microscopic social network evolution, show that it has powerlaw asymptotics¹

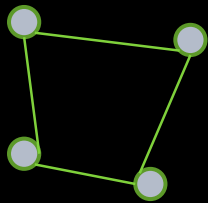
¹ Leskovec et al. KDD 2008

Evolving large networks:

Types of data

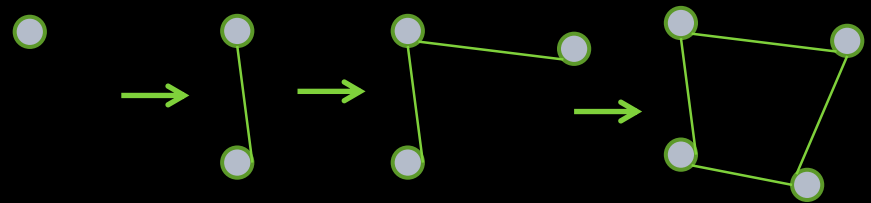
Snapshot

- At one point in time



Temporal data

- A log of the evolution
- hard: nobody logged all events in the creation of the internet up to now



Temporal models:

Types of data

- Snapshot:
 - Historical information may be missing
- Can we still detect traces of evolution in the network?
 - Sometimes yes, e.g. phylogenetic trees

Temporal models: Summary

- Scale of evolution:

- local
- global
- asymptotical

- Types of data:

- Snapshot
- Temporal

Temporal data:

Open problems

- Integration of local and global/asymptotic levels?
- Can we learn dynamics from snapshots?

Conclusions

- Several domains study networks from different points of view (and can learn from each other)
- This tutorial: local level
 - Others: global level
 - Progress towards integration

Questions?

?